Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA DE COMPUTADORES

Trabajo Fin de Grado

Estudio de gráficas de tabletas digitalizadoras

Autor: Héctor Hernández Cristóbal

Tutor/es: Antonio García Herraiz

2020-2021

Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA DE COMPUTADORES

Trabajo Fin de Grado

Estudio de gráficas de tabletas digitalizadoras

Autor: Héctor Hernández Cristóbal

Titulación: Grado en Ingeniería de Computadores

Código: 15/2021/TFG

Tribunal calificador:

Presidente: Bernardo Alarcos Alcázar

Vocal 1º: Juan Ramón Velasco Pérez

Vocal 2º: Antonio García Herráiz

Fecha: Julio 2021

Agradecimientos a mis padres y a mis amigos
por haber estado cuando más les necesitaba

ÍNDICE

Resumen

Partiendo del hecho de tener que analizar las pruebas VMI manualmente, llevando tediosas tareas de organización y gran cantidad de documentos mediante esta aplicación he conseguido solucionarlo generando ficheros Excel para cada una de las pruebas VMI realizadas, correspondiendo con cada niño y mostrando ciertos valores útiles para el psicólogo.

Palabras Clave

Test Beery, Niños, Python, Wacom, Psicología

Abstract

Starting from the fact of having to analyze the VMI tests manually, carrying out tedious organizational tasks and a large number of documents through this application, I have managed to solve it by generating Excel files for each of the VMI tests carried out, corresponding to each child and showing certain useful values for the psychologist.

Index terms

Beery Test, kids, Python, Wacom, psychology

Resumen Extendido

Hoy en día los profesionales que trabajan con niños para detectar problemas psicológicos y psicomotrices deben pasar por largas etapas de gestión de documentos, análisis de resultados y aumento de pruebas por falta de datos en los resultados obtenidos tras realizar pruebas. En este trabajo, buscando una solución a dicho problema, se plantea una aplicación software encargada de facilitar la tarea del profesional de detectar en niños dificultades relacionadas con la habilidad motriz de dibujar sobre un papel basándose en la prueba VMI.

Mediante una aplicación *Python* se ha conseguido obtener una serie de estadísticos característicos guardados en un fichero Excel a partir de los dibujos realizados por el niño sobre una tableta gráfica Wacom Bamboo Slate. En este fichero se encuentran parámetros útiles para realizar un análisis más preciso por parte de los profesionales y así conseguir reducir el número de pruebas a realizar a los alumnos por falta de conocimiento o por ambigüedad en alguno de las puntuaciones obtenidas mediante la escala de Beery.

Mediante la creación de ficheros Excel se facilita la gestión de las evaluaciones realizadas a las personas, teniendo dentro de una misma carpeta todos estos archivos con un formato fácil de comprender y gestionar, dejando agrupados los ficheros correspondientes a la misma persona, pudiendo observar fácilmente una posible evolución positiva o negativa. Además se han planteado todos los problemas que se podrían producir por el hecho de estar trabajando con niños, teniéndolo en cuenta a la hora del desarrollo con el fin de hacer una aplicación lo más robusta posible, consiguiendo que aunque el niño se salga de la línea asignada para el dibujo, realice un número mayor o menor de líneas o cambie la orientación del dibujo entre otras posibilidades, la aplicación siga funcionando correctamente relacionando cada dibujo con el ejercicio correspondiente, evitando tener que rehacer la prueba si el alumno no ha realizado correctamente los dibujos, resultando en un mayor tiempo invertido por el profesional, retardando el diagnóstico de la persona que está siendo evaluada por la prueba VMI.

La aplicación resultante está pensada para ser usada por cualquier persona, aunque no posea conocimientos de informática, llevando la ejecución del software a lo más automático posible, permitiendo dejar el ordenador procesando los datos mientras el profesional dedica su tiempo a realizar el análisis visual de los dibujos, aumentando la productividad.

El programa gestiona los datos recibidos y los muestra en forma de gráficas, ayudando a comprender mejor los resultados obtenidos. Además, el Excel contiene tablas relacionadas con cada dibujo realizado, consiguiendo un análisis más profundo y una representación más visual de lo que sería una lista de valores, permitiendo estudiar ejercicio por ejercicio todos los parámetros necesarios en profundidad.

1. Introducción

El correcto desarrollo de las habilidades es uno de los aspectos que más se enfatizan durante los primeros años de los niños en el colegio, por ello realizar ciertas pruebas en estas edades puede facilitar la detección de problemas que sufrirán a lo largo de su vida si no son corregidos.

Basándose en el estudio de 1992 de McHale y Cermak se muestra que "los niños de las escuelas públicas intervienen entre un 30% a un 60% del día en la realización de tareas motoras finas", empleando en la mayoría de ellas lápiz y papel. La destreza de escritura manual puede derivar en diversos componentes interesantes para el estudio de profesionales de diversas áreas como psicólogos, terapeutas ocupacionales, especialistas en problemas de aprendizaje y otros, además del maestro.

Según Hsu (1997), una de las herramientas de evaluación estandarizada más populares y usada internacionalmente es la Developmental Test of Visual Motor Integration, conocida como Beery-Buktenica Developmental Test of Visual Motor Integration (Beery-VMI). Esta prueba consiste en 24 dibujos de figuras geométricas colocados en orden ascendente según su dificultad, el niño evaluado deberá copiar dichas figuras en un espacio reservado bajo cada una, sin límite de tiempo. Inicialmente fue pensada para niños entre 2 y 15 años, pero se adapta más para ser usada con niños de preescolar y primaria.

Se ha realizado mucha investigación referida a la gran importancia que tiene la relación existente entre experiencias tempranas, inteligencia y adquisición escolar (demostrado por Piaget, 1956 y Hunt, 1961). Dichas funciones deben ser practicadas en conjunto y no por separado, por ejemplo, para poder escribir, un niño debe coordinar percepción visual y coordinación motriz. Por lo tanto, el trabajo debe ser enfocado en estas áreas.

Aunque el VMI es usado con fines diagnósticos, es de esperar que los maestros de niños normales o inhábiles encuentren en esta prueba un instrumento útil para su trabajo.

Es importante recalcar que hay una diferencia entre fines educativos y diagnóstico clínico, no diferenciarlo conduciría al fracaso, si un maestro tomase VMI como una prueba que mide la inteligencia o maduración neurológica y al aplicarlo uno de los niños fracasara podría suceder lo siguiente:

a.- El maestro recomienda a los padres sacar al niño y esperar un año para que "madure".

b.- El maestro puede sentirse incapaz de enseñarle al niño porque el problema escapa a su conocimiento.

c.- El maestro podría diagnosticar por su cuenta e informar a los padres que su hijo es retardado o lesionado.

Cualquiera de estas posibilidades sería antiético, la función de los educadores es enseñar y actuar en forma positiva, alarmar a los padres no es conveniente.

La forma de aplicar la prueba es la siguiente:

-El alumno deberá copiar las figuras con el lápiz, sin borrar ni remarcar, respetando el orden y realizando solo un intento por figura.

-Colocar el cuadernillo frente al niño y cuidar que no vea las figuras más difíciles antes de tiempo.

-El cuadernillo y el niño deben situarse en el centro de la mesa.

-Abrir el cuadernillo en la primera página de figuras e inmediatamente mientras se señala la figura el preguntar al niño "¿Puedes hacer uno igual?". Dejar que el alumno responda y luego señalar el espacio debajo de la figura y decir "Haz tu figura aquí". Si es necesario incentivar al alumno (Bajo ningún punto de vista pueden el examinador o el niño pasar su dedo encima de la figura o con el lápiz. No dar nombres a las figuras. Ejemplo: No decirle "Haz el circulo" o "Dibuja la pelota").

-Ayudar al niño hasta que comprenda lo que el examinador espera que realice. "Sigue copiando las figuras y cuando termines, da la vuelta a la hoja". El niño no debe sentirse presionado por tomar tiempo o por escribir observaciones.

-Si el alumno no comprende el mecanismo o falla en los tres primeros intentos, se puede volver la carilla de la primera hoja y utilizar las figuras punteadas como práctica. El examinador pasará por los puntos y luego invitará al alumno a que lo haga. Si el alumno logra lo pedido dejarlo que intente copiar figuras.

-Si el alumno ha fallado en tres figuras consecutivas se puede suspender la prueba. Si se desea se puede continuar para observar como el niño se enfrenta a las figuras más complicadas (según el autor, sus experiencias le dicen que los niños no se desaniman frente a sus fracasos y que incluso intentan otras figuras).

Para cada elemento del VMI hay una hoja de información llamada "criterio de evaluación", lo cual da una relación de edad y los requisitos para pasar de grado.1
Debido a esto, el uso de pruebas estandarizadas exige un alto sentido de responsabilidad por parte de los profesionales que las utilizan, especialmente a la hora de interpretar los resultados. Aunque Beery-VMI conste de dos subpruebas además de la inicial, los investigadores tienden a usar sólo la prueba principal en sus estudios.2
En este trabajo se propone una digitalización del proceso para facilitar el estudio y permitir analizar parámetros nuevos que anteriormente eran imposibles de tener en cuenta (tiempo empleado, exactitud entre el dibujo original y el realizado por el niño, velocidad y aceleraciones, los segmentos del dibujo donde ha tardado más, fuerzas ejercidas, punto de inicio, número de trazos,

entre otros) mediante programas en Python se evaluarán estos ejercicios y se obtendrá un Excel donde se encontrarán todos los valores. La forma de obtener los dibujos será mediante una tableta digitalizadora de la marca Wacom, siendo el modelo exacto Bamboo Slate, este dispositivo tiene la característica de capturar datos de lo dibujado en un papel colocado sobre él, evitando así posibles distracciones o problemas de falta de costumbre (otras tabletas cogen los dibujos sobre una pantalla, al estar acostumbrados a dibujar sobre papel, si se utilizan esas se podría obtener un resultado distinto).

La forma en la que se propone relacionar los dibujos con cada persona y seguir su progresión en el tiempo consiste en la realización de un cuaderno que contenga todas las pruebas y un Excel con todos los datos obtenidos en cada fecha, teniendo agrupados los ejercicios y los parámetros para así poderlo visualizar de forma más cómoda.

Para la realización del estudio VMI por los psicólogos se empleará una variante de la prueba original, utilizando modelos nuevos de figuras y una disposición diferente, el objetivo final es realizar los mismos ejercicios a los mismos niños en diferentes periodos de tiempo, todo esto se recogerá en un cuadernillo donde estarán las diferentes pruebas realizadas y pendientes por hacer, con la finalidad de ver la evolución del niño a lo largo de su desarrollo y detectar posibles problemas psicológicos o motrices que puedan afectar a lo largo de su vida.

Los ejercicios serán los siguientes con el formato que se ve, tres dibujos y tres huecos para que el niño replique el enunciado.

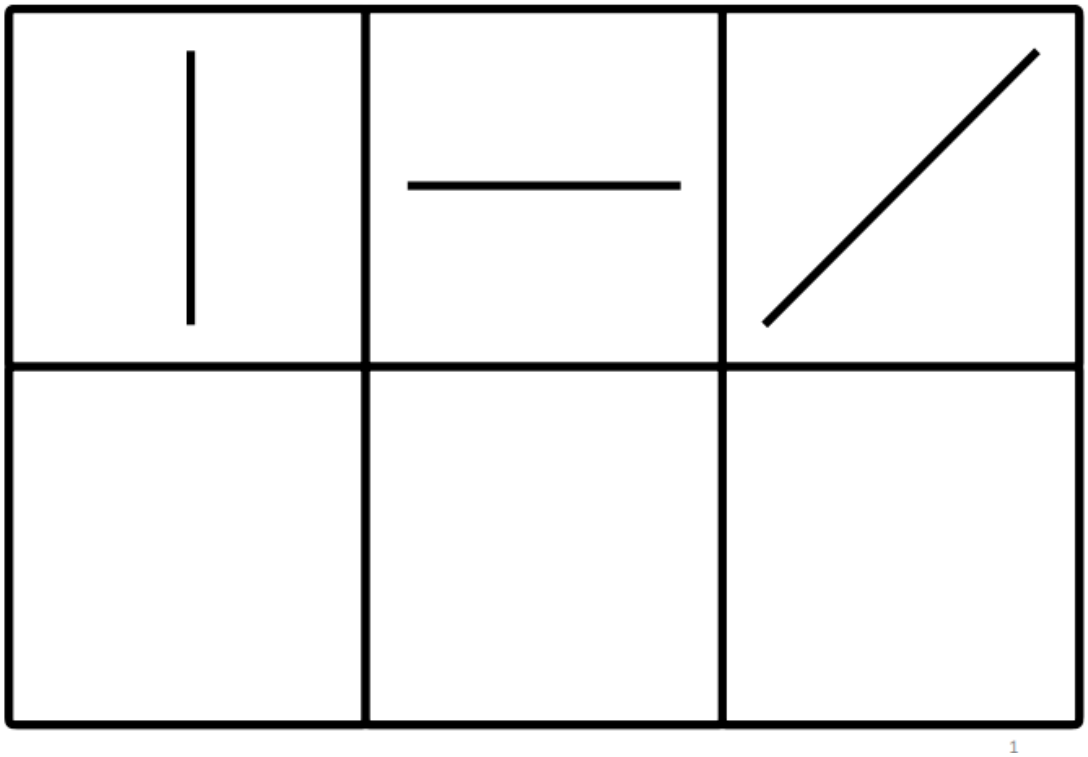


Ilustración 1 Hoja 1 Cuadernillo

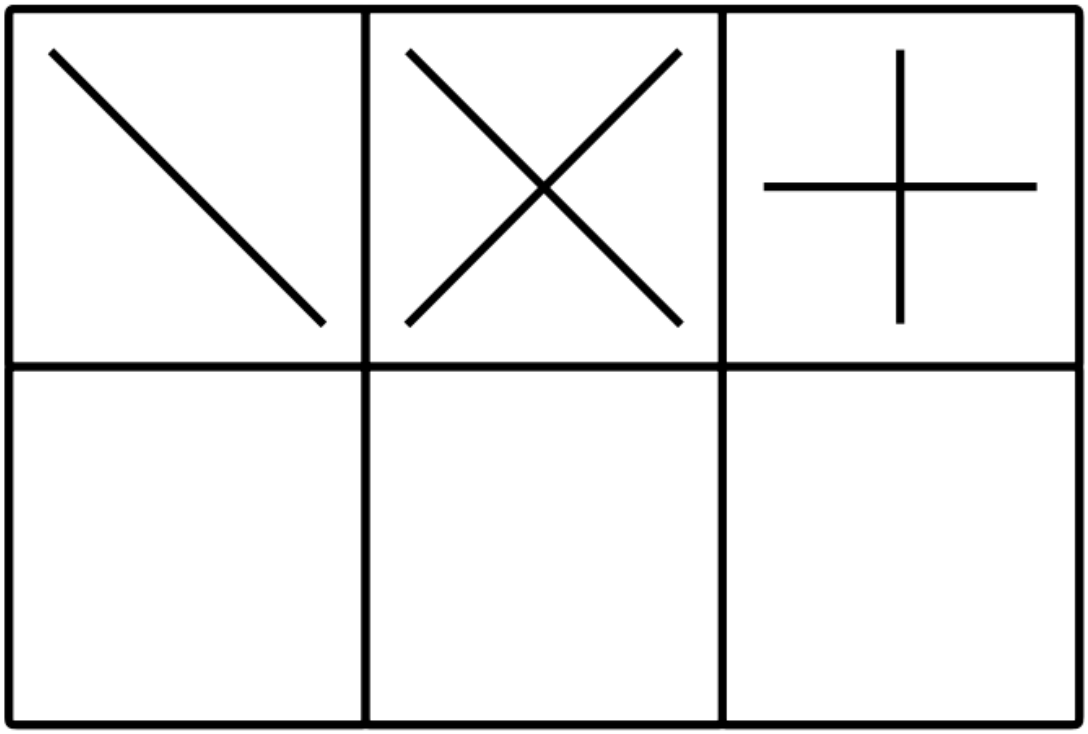
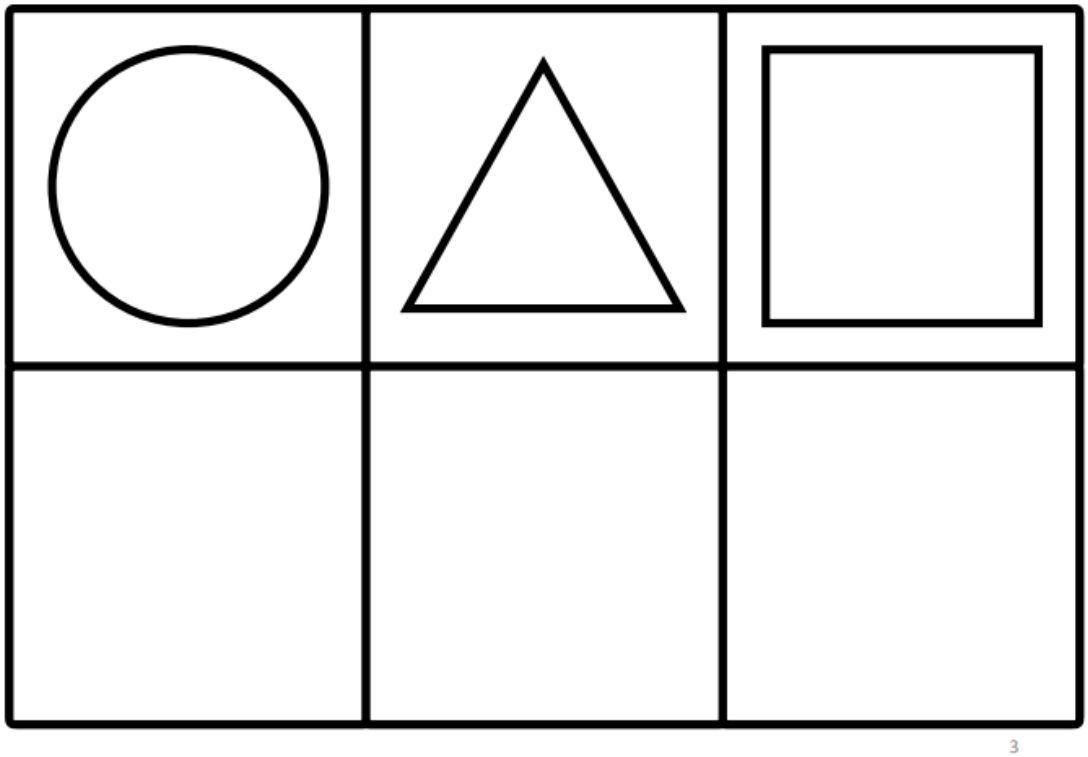


Ilustración 2 Hoja 2 Cuadernillo



3

Ilustración 3 Hoja 3 Cuadernillo

En caso de los siguientes ejercicios, el niño deberá dibujar dentro de los laberintos sin salirse de las líneas delimitadoras:

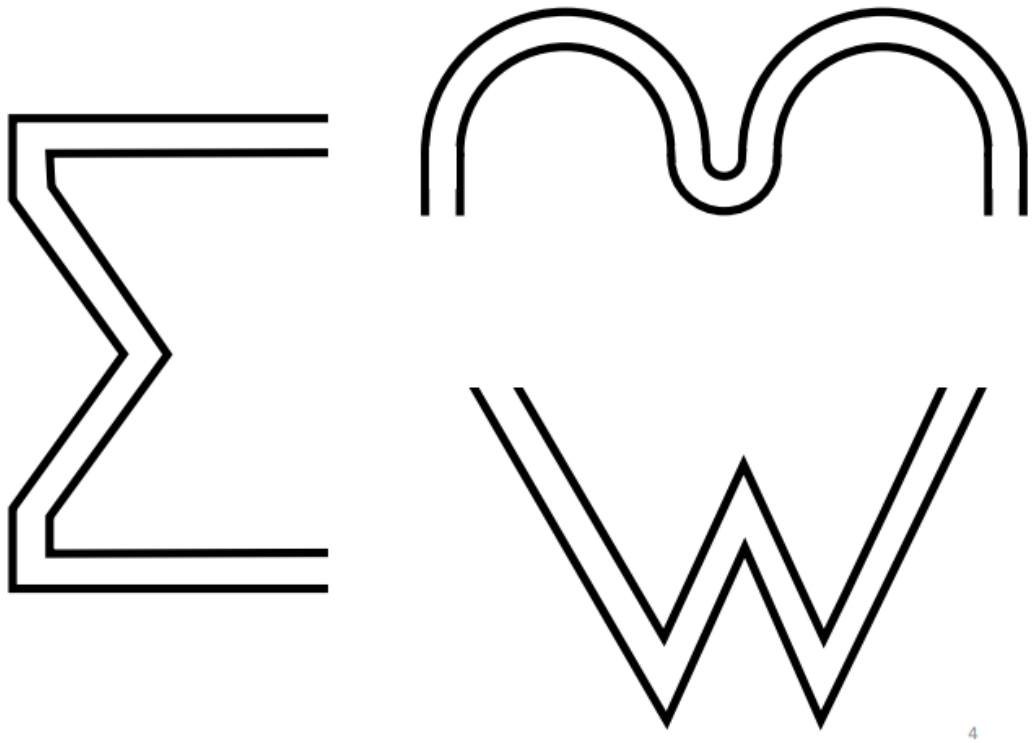


Ilustración 4 Hoja 4 Cuadernillo

El resultado final podrá ser observado fácilmente desde el cuaderno donde se encontrarán los dibujos, no siendo necesaria la implementación de su evaluación en el programa, teniendo únicamente en cuenta los parámetros que no se pueden ver a simple vista en el papel.

Lo primero que se necesita para realizar este trabajo es la traducción de los datos obtenidos en la tableta digitalizadora a un sistema reconocible por Python, mediante *Inkspace* conseguiremos descargar en el ordenador el dibujo como *yaml*, para facilitar la lectura de los datos, lo convertiremos a *json*, mediante el programa *pywillparser3*.

Una vez tengamos ya los datos en un formato cómodo para trabajar, deberemos cargarlos en el programa Python para su posterior evaluación, lo conseguiremos usando el paquete *json*, el cual nos permite acceder a cada uno de los puntos que representan el dibujo y las fuerzas que se han ejercido a la hora de dibujarlo.

Con los datos ya en el programa Python deberemos analizarlos en función del ejercicio al que correspondan, teniendo en cuenta la circularidad, lo rectas que son las líneas, la orientación del dibujo, etc. Para ello deberemos crearlos específicamente para cada uno de ellos, en función de las especificaciones de los especialistas.

Una vez evaluado el ejercicio procederemos a almacenar todos los parámetros importantes obtenidos en un Excel, para facilitar su estudio a los profesionales.
Los objetivos de este trabajo son obtener un conjunto de datos correspondientes a los dibujos de los niños con el fin de detectar posibles problemas mentales o psicomotrices basándonos en los criterios del VMI.
A lo largo del documento se verán diversos campos correspondientes a la aplicación desarrollada, en primer lugar, se verá una descripción del trabajo, comenzando por la base teórica, donde se analizará de forma matemática "sobre el papel" todos los cálculos que estarán presentes en el software, mostrando todos los resultados que se obtendrían de cada problema que se encuentra. Tras esto se desarrollará la descripción experimental, explicando el software que se creará para resolver el problema actual, además de la inserción de líneas de código correspondientes a cada clase.
A continuación, se mostrará la arquitectura de la aplicación, junto a un esquema de clases donde se mostrarán los métodos implicados en cada una de las clases y los parámetros que relacionan unas clases con otras, además de la utilidad que cada una de ellas tendrá.
Después, se verá el presupuesto, donde se indica el tiempo que he invertido en el desarrollo de la aplicación mediante un diagrama de Gantt, y el dinero necesario para ejecutar esta aplicación en un ambiente escolar, con el fin de realizar la prueba a un conjunto de alumnos.
Finalmente, se encontrarán las conclusiones obtenidas tras la realización del trabajo, así como el trabajo futuro que queda restante. Tras esto estará la bibliografía, y, dentro de los anexos, el manual de usuario explicando el funcionamiento del software.

2. Descripción del trabajo

Base teórica

El desarrollo del trabajo consiste en el análisis gráfico de diferentes dibujos, obteniendo varios parámetros, cada uno de ellos tiene un estudio matemático para obtener la forma de implementarlo dentro del entorno de Python, además de ver las adaptaciones necesarias para el formato de datos que se maneja dentro de la aplicación.

Carga de datos

Al consistir en la carga de un archivo *JSON*, los ejercicios se almacenan en dos campos distintos, *paths* y *strokes*, siendo el primero el correspondiente a todos los puntos dibujados y el segundo a las fuerzas ejercidas para cada punto. Es posible conocer el número de líneas que ha dibujado contando los *paths*. Además, en caso de que alguno de los puntos se repita, podemos saber si el niño dejó el bolígrafo parado sobre la tableta en algún momento, obteniendo el número de paradas.

```
[{
        “paths”: [
          {
                “points”: [
                        [x1, y1],
                        [x2, y2],
                        …
                        [xn, yn]
                ],
                “strokes”: [
                        S1,
                        S2,
                        …
                        Sn
],
“avg_width”: m,
“color”: [
        255
]
}
        ],
        "width": "864.0",
       "height": "592.0"
        }]
```

El esquema anterior es el correspondiente con un fichero *json* generado basándose en la tableta Bamboo Slate, al final se encuentra *width* y *height* (ancho y alto) de la superficie que admite la entrada de dibujos del hardware. En "*paths*" se verán las líneas dibujadas, conteniendo un vector de "*points*" y "*strokes*" para cada trazo, en estos se guardarán todos los n puntos dibujados cada 0.005 segundos (200 Hz de frecuencia), y la fuerza ejercida para cada uno de esos puntos (siendo el mínimo 1 y el máximo 2), respectivamente. Al final nos encontraremos el parámetro color correspondiente a la escala RGB del color seleccionado para realizar el dibujo (en otras tabletas de la marca Wacom es posible dibujar en diferentes colores y debido a que comparten software está implementado para todas ellas) y previamente el parámetro *av_width*, el cual indica la media de las presiones ejercidas en dicho trazo.
El fichero se obtiene del programa *PywillParser*, a partir del archivo *yaml* generado desde la aplicación de Wacom.

Regresión Lineal

Las líneas realizadas por los niños no son líneas rectas, evaluar aspectos como la inclinación de la línea puede producir un resultado muy diferente al real, dependiendo de qué puntos se elijan para calcularla:

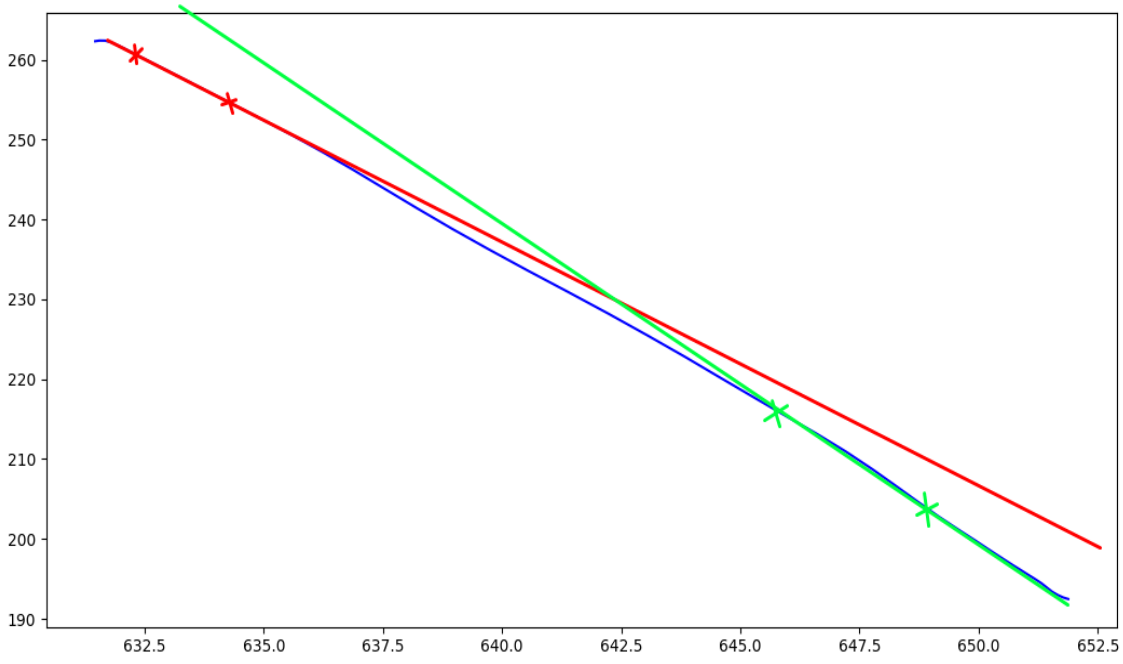


Ilustración 5 Error en la toma de línea

Siendo la línea azul la dibujada por el niño y las rectas verde y roja corresponden a la obtención de ángulos según qué puntos de ésta se utilicen para calcularlo, viendo cómo el resultado varía mucho en función de qué parte de la recta se elija para obtenerla, debiendo buscar una forma más general y precisa de obtener la recta que más se asemeje a la dibujada.
La solución a este problema es calcular la regresión lineal para obtener una recta aproximada al dibujo realizado por el niño, pudiendo calcular así el ángulo de inclinación de forma más precisa y ver las diferencias existentes entre la original y la que de verdad está dibujada en el papel.
En estadística, la regresión lineal (también conocida como ajuste lineal), es un modelo matemático empleado para aproximar la relación de dependencia entre una variable dependiente, m variables independientes y un término aleatorio. Está basada en el método de los mínimos cuadrados (Legendre, 1805), desarrollado más en profundidad por Gauss. Existen varios tipos de modelos de regresión lineal, para el estudio de las rectas se emplea la regresión lineal simple, el cual está formado por dos variables estadísticas que se relacionan funcionalmente.4

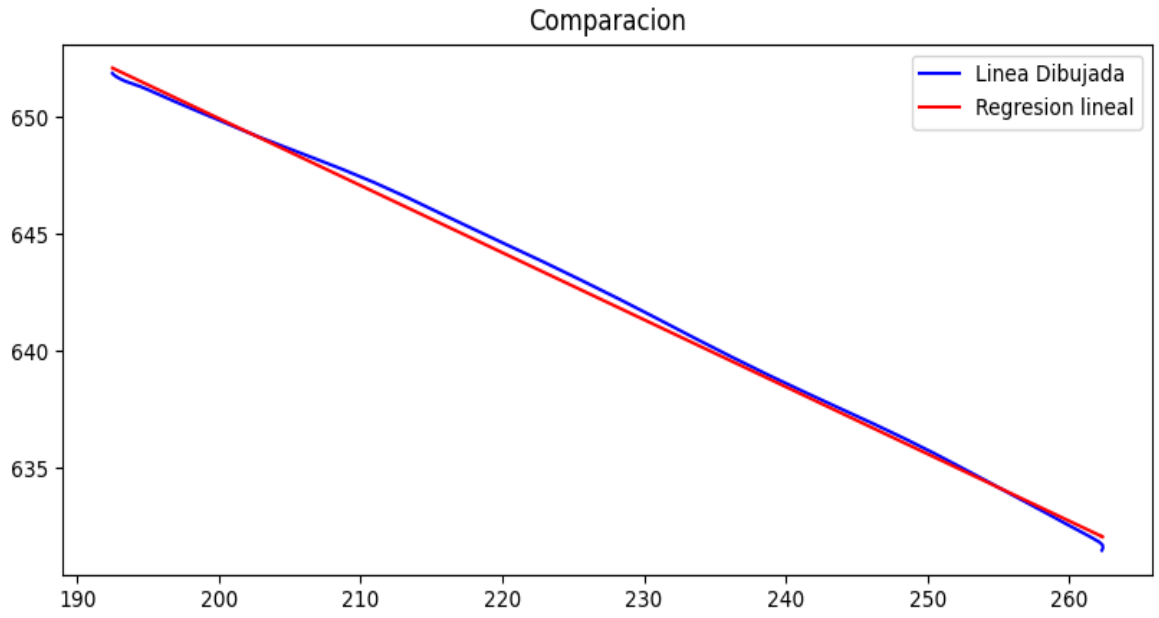


Ilustración 6 Regresión Lineal

La forma matemática de calcular la regresión es relacionando la variable dependiente Y con m variables regresoras Xj con j = 1, 2, …, m:

$$Y = \beta y = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_n + e = \beta_0 + \sum_{j=0}^{m} \beta_j x_j + e$$

Siendo *e* una variable aleatoria que recoge todos aquellos factores de la realidad no controlables u observables y que por tanto se asocian con el azar.

Pendiente de la recta

Una vez calculada la recta regresiva se podrá calcular la pendiente, el problema de tratar de calcularla para la recta original es que, como se ha visto anteriormente, depende de los puntos que se escojan, al no pertenecer a una recta la pendiente variará demasiado.
Para calcularla se han de coger dos puntos P_1 y P_2 que pertenezcan a la línea. Con ellos se calcula la pendiente (inclinación de la recta con respecto al eje de abscisas):

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

La pendiente se puede utilizar para calcular el ángulo, y así ver si la línea dibujada por el niño se corresponde con la que se pedía en el enunciado VMI. Se calculará mediante el arco tangente de la pendiente.

$$\alpha = arctg(m)$$

Siendo *m* la pendiente de la recta que se está evaluando.

Obtención del punto de inicio

Obteniendo el primer punto dibujado en cada línea y ejercicio es posible conocer desde dónde tiende a dibujar el niño para cada tipo de ejercicio, para ello se compararán las x y las y del primer y último punto de cada recta, obteniendo una lista de posiciones relativas, indicando si prefiere dibujar de izquierda a derecha o de arriba abajo para cada uno de los problemas que se le planteen.

$$\forall x_1 > x_2 \;\; Empieza\ por\ la\ derecha$$
$$\forall x_1 < x_2 \;\; Empieza\ por\ la\ izquierda$$
$$\forall y_1 > y_2 \;\; Empieza\ por\ arriba$$
$$\forall y_2 > y_1 \; Empieza\ por\ abajo$$

Tiempo empleado

Como la frecuencia de muestreo de la tableta gráfica Bamboo Slate es conocida (200 Hz), podemos obtener los tiempos correspondientes a cada línea, aunque no se podrán obtener los valores de tiempos intermedios, ya que no cuenta de un reloj interno. Para calcular el tiempo emplearemos la fórmula de la frecuencia:

$$t = \frac{numM}{f}$$

Siendo *t* el tiempo, *numM* el número de muestras y *f* la frecuencia a la que se muestrea.

Distancia dibujada

Una vez se tenga el tiempo, se puede calcular la distancia invertida en cada cuarto de tiempo del dibujo, ver velocidades y aceleraciones, en qué punto va más rápido y en cuál más lento:

Ilustración 7 Cuadrantes en línea dibujada

Para calcular la distancia se dividirá el tiempo total entre 4 y se calculará para cada cuadrante:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Siendo *x1* e *y1* el primer punto y *x2* e *y2* el último punto.
Con estas distancias se podrá ver en el segmento en el que ha ido más deprisa y en el que ha tardado más en realizar el dibujo, comparando los valores obtenidos.

Velocidad

Teniendo la distancia de la recta dibujada es posible calcular la velocidad empleada para cada segmento, en este caso, al igual que para la distancia, se tomará en cuatro segmentos distintos, la forma de conocer la velocidad es empleando el número de puntos de dicho cuadrante y la frecuencia de muestreo.
Lo primero es obtener el tiempo empleado a la hora de dibujar la línea, esto se realizará dividiendo el número de puntos entre la frecuencia.
Una vez hecho esto, se debe calcular la distancia entre el primer punto y el último del segmento.
Finalmente se tiene distancia d y tiempo t, pudiendo calcular tras esto la velocidad:

$$v = d * t$$

Aceleración

La aceleración es la variación de velocidad por unidad de tiempo, será útil a la hora de ver en qué partes del dibujo el alumno ha aumentado o disminuido la velocidad, se calcula empleando la siguiente fórmula:

$$a = \frac{d_1 - d_2}{t_1 - t_2}$$

Circularidad

A la hora de analizar el dibujo de la circunferencia realizado por el niño, se tendrá en cuenta la circularidad de esta. Esto se consigue empleando los momentos de *Hu*, en *OpenCV*. Son una cantidad, una cifra que captura información básica de la forma de un objeto en una imagen, como su área, coordenadas de su centroide, orientación, entre otras propiedades. Es una expectativa estadística de una variable aleatoria, es decir, un valor que se espera que la variable tome en un futuro.
Ming-Kuei Hu (1962), postuló siete valores (momentos) que pueden usarse para sintetizar la forma de un objeto en una imagen, afirmando en su publicación Visual Pattern Recognition que dichos valores no se ven afectados por cambios de rotación, translación, escala y reflexión de una imagen u objeto.
Para calcular los siete momentos de Hu se puede trabajar con la imagen binaria, segmentada, del objeto de interés o con su contorno.[5]

En esta ocasión solo interesa conocer la circularidad, por lo que se empleará la siguiente fórmula:

$$F = \frac{4\Pi A}{P^2}$$

Siendo *A* el área del circulo:

$$A = \pi r^2$$

Con *r*, el radio, y *P* la fórmula del perímetro:

$$P = 2\Pi r$$

El rango del resultado obtenido en F va de 0 a 1, siendo 1 la circunferencia perfecta.

$$F \in \mathbb{R}\{[0,1]\}$$

Área y perímetro

Para las figuras geométricas con más de un lado (triángulo, círculo y cuadrado), será posible realizar el cálculo del área y del perímetro. Gráficamente se puede definir el perímetro de una figura como la suma de todos sus lados, y el área como la superficie de dicha figura. De la forma en la que se almacenan los datos es difícil conocer cuándo un punto pertenece a un lado u a otro, ya que, debido a una vibración en el trazo del niño, si comparamos las x y las y para obtener los lados, podemos obtener un falso positivo a la hora de su detección.
La forma en la que se obtienen estos parámetros es calculando segmentos de pares *x* e *y*, mapeándolos y guardándolos en un arreglo para posteriormente usarlos (función zip de Python).
El perímetro se calcula con la siguiente fórmula:

$$P = \left| \Sigma_{s<n}^{s=0} \sqrt{(x_0 - x_1)_s^2 + \overline{(y_0 - y_1)_s^2}} \right|$$

Con *s* siendo el número de segmentos calculados anteriormente, yendo desde *0* hasta *n*, y los valores de *x* e *y* los correspondientes al primer y último punto de cada segmento.
Para el área se utiliza la siguiente fórmula:

$$A = \frac{|\sum x_0 y_1 - x_1 y_0|}{2}$$

Al igual que con el perímetro, se calcula para todos los segmentos, con el primer y último punto, sumándolos y calculando el valor absoluto.

Centroide

Para las figuras geométricas planas es posible calcular el centroide (intersección de todos los hiperplanos que dividen al objeto en dos partes de volumen 2 con respecto al hiperplano).
El centroide corresponde con el centro de simetría de una figura geométrica. Se calcula con la siguiente fórmula:

$$centroide = \sum \frac{x}{n}, \sum \frac{y}{n}$$

Es decir, sumando todos los valores de *x* y dividiéndolos entre el número de valores (número de puntos), haciendo lo mismo para la *y*, se obtienen las coordenadas correspondientes a dicho punto.

Presiones

Como se ha visto anteriormente, el archivo *json* obtenido de la exportación de los datos del dibujo del niño contiene un apartado de "*strokes*", aquí se guardan las fuerzas ejercidas por cada punto, por lo que, hay el mismo número de fuerzas que de puntos. Estas presiones se miden en un rango entre 1 y 2, siendo 1 la mínima presión y 2 el máximo que se puede apretar.
Conocer cuánto aprieta el niño servirá para ver posibles problemas psicológicos o motrices que pueda presentar el alumno a la hora de su desarrollo. Gracias a la tableta gráfica se puede medir esto, mientras que con la forma tradicional sería imposible cuantificar este parámetro, únicamente permitiría conocer por el grosor del trazo realizado.
Sobre estas presiones se calculará la media y los puntos en los que más y menos fuerza se ha ejercido, en función de cuadrantes de tiempo, igual que se ha realizado anteriormente para obtener los puntos en los que ha ido más deprisa y más despacio, pudiendo relacionar velocidad y presión al final.

Almacenamiento de datos en un archivo Excel

Con el objetivo de facilitar la gestión de los datos recogidos y su posterior análisis por parte del profesional, todos los resultados son almacenados en un fichero Excel, conteniendo los puntos captados por la tableta [x, y], la presión ejercida en cada punto, la distancia entre el punto anterior y el actual, la velocidad y aceleración por cada par de puntos.
Además, se almacena una tabla con los datos devueltos por el programa tras el análisis y tres gráficas por cada trazo que corresponden con la presión ejercida, la velocidad y la aceleración en

ese segmento. Todo esto se incluirá en una hoja correspondiente a cada dibujo que haya realizado el niño, en caso de que uno de los dibujos no haya sido rellenado, la hoja quedará en blanco.
En la primera hoja se almacenan todas las tablas de cada uno de los ejercicios con el objetivo de dar una visión general de todos los dibujos sin entrar a detalles de puntos, gráficas y valores de aceleración y velocidad, permitiendo hacer una comparación de dos o más ejercicios de forma más cómoda.
El funcionamiento de *openpyxl* se basa en la selección de hojas dentro de un fichero Excel, para seleccionar el archivo se emplea "*wb*", el cual tiene la opción de crear un fichero nuevo (*wb = openpyxl.Workbook()*) o elegir un Excel ya creado (*wb = openpyxl.load_workbook('sample.xlsx')*)[6].
Openpyxl está pensado para trabajar accediendo a las celdas, pudiendo ser mediante el nombre ("*A2*", "*B7*", etc), o empleando la opción de filas y columnas (*cell(row = j, column = 11)*), tras esto se añade ".*value*" para indicar el valor que va a tener dicha celda.
Los valores de puntos, distancia, velocidad, aceleración y presión son añadidos directamente del array, empleando una posición fija de columna para cada valor y un valor de fila autoincrementada, con el fin de crear una tabla.

Lo mismo es aplicado a los valores obtenidos por el estudio de la aplicación, almacenando en una columna los nombres de los resultados que se guardan en la columna siguiente.
Con *Openpyxl*, también es posible incluir gráficas basándose en puntos ya existentes en el fichero, se debe importar *linechart* para dibujar la gráfica de forma lineal, también es posible utilizar gráficos de barras con *barchart*. Se indican las columnas que se van a incluir en la gráfica, las filas que contendrá y la celda en la que se quiere insertar. [7]

Para facilitar la visualización de los datos de las tablas se puede incluir color de fondo a las celdas e insertar bordes sólidos, los cuales diferenciarán mejor unos valores de otros y será más cómoda la utilización. Se han de importar "*Color*", "*PatternFill*" y "*Border*". Además, deben ser ajustadas las columnas al tamaño de los valores que se incluyen, para ver todo sin tener que modificarlo manualmente. [8]

Un ejemplo de la hoja de un ejercicio de un alumno tiene el siguiente formato:

- Los valores de los puntos, presiones, distancias, velocidades y aceleraciones se almacenan entre la columna A y la G:

A	B	C	D	E	F	G
Trazo 1	(269 muestras)					
	X	Y	Pres	Dist (pts)	Vel cm/s	Acel cm/s2
1	419,60	127,24	1,00	0,00	0,00	0,00
2	419,60	127,24	1,00	0,00	0,00	0,00
3	419,56	127,24	1,06	0,04	0,20	40,00
4	419,42	127,26	1,19	0,14	0,71	101,42
5	419,13	127,31	1,36	0,29	1,47	152,86
6	418,71	127,42	1,49	0,43	2,17	139,89
7	418,23	127,58	1,54	0,51	2,53	71,80
8	417,73	127,77	1,54	0,53	2,67	28,92
9	417,25	127,97	1,53	0,52	2,60	-14,88
10	416,81	128,15	1,53	0,48	2,38	-44,61
11	416,38	128,29	1,54	0,45	2,26	-23,18
12	415,92	128,41	1,55	0,48	2,38	23,18
13	415,40	128,49	1,56	0,53	2,63	50,72
14	414,84	128,53	1,57	0,56	2,81	35,31
15	414,29	128,53	1,58	0,55	2,75	-11,43
16	413,77	128,52	1,58	0,52	2,60	-29,90
17	413,29	128,49	1,59	0,48	2,40	-39,16
18	412,80	128,46	1,60	0,49	2,45	9,98
19	412,29	128,43	1,60	0,51	2,55	19,96

Ilustración 8 Puntos, presiones, distancia, aceleración y velocidad por trazo

- En las columnas J y K se establecen los valores de lo estudiado en la aplicación en formato de tabla, con bordes sólidos y color de fondo:

I	J	K
	ejercicio	1
	Angulo recta	2,48
	Num Puntos	269
	Numero trazos	1
	Tiempo (s)	1,345
	distancia en el cuarto 1 (cm)	1,165775
	distancia en el cuarto 2 (cm)	1,66865
	distancia en el cuarto 3 (cm)	1,980225
	distancia en el cuarto 4 (cm)	0,740025
	distancia total (cm)	5,554675
	ha ido más rápido	3
	ha ido más lento	4
	velocidad media	4,145279851
	velocidad en el cuarto 1 (cm/s)	3,479925373
	velocidad en el cuarto 2 (cm/s)	4,981044776
	velocidad en el cuarto 3 (cm/s)	5,911119403
	velocidad en el cuarto 4 (cm/s)	2,209029851
	aceleracion media	1,648529739
	aceleracion en el cuarto 1 (cm/s)	10,38783693
	aceleracion en el cuarto 2 (cm/s)	4,480953442
	aceleracion en el cuarto 3 (cm/s)	2,77634217
	aceleracion en el cuarto 4 (cm/s)	-11,05101359
	punto inicio	derecha arriba
	Pendiente Recta	0,043276937
	fuerza media al inicio	1,618507463
	fuerza media en el segundo	1,759701493
	fuerza media en el tercero	1,849552239
	fuerza media al final	1,84880597
	media Fuerza	1,763

Ilustración 9 Valores obtenidos por la aplicación en el dibujo

- Por último, se encuentran las gráficas de presión, velocidad y aceleración, a la derecha de la tabla vista en la ilustración anterior:

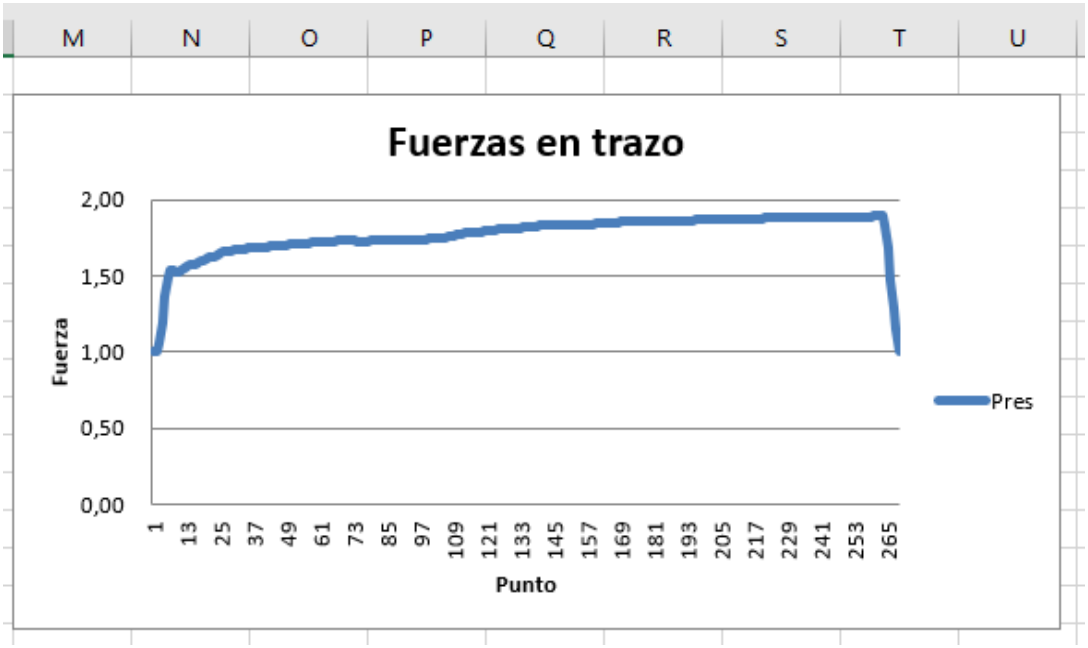


Ilustración 10 Gráfica de fuerzas

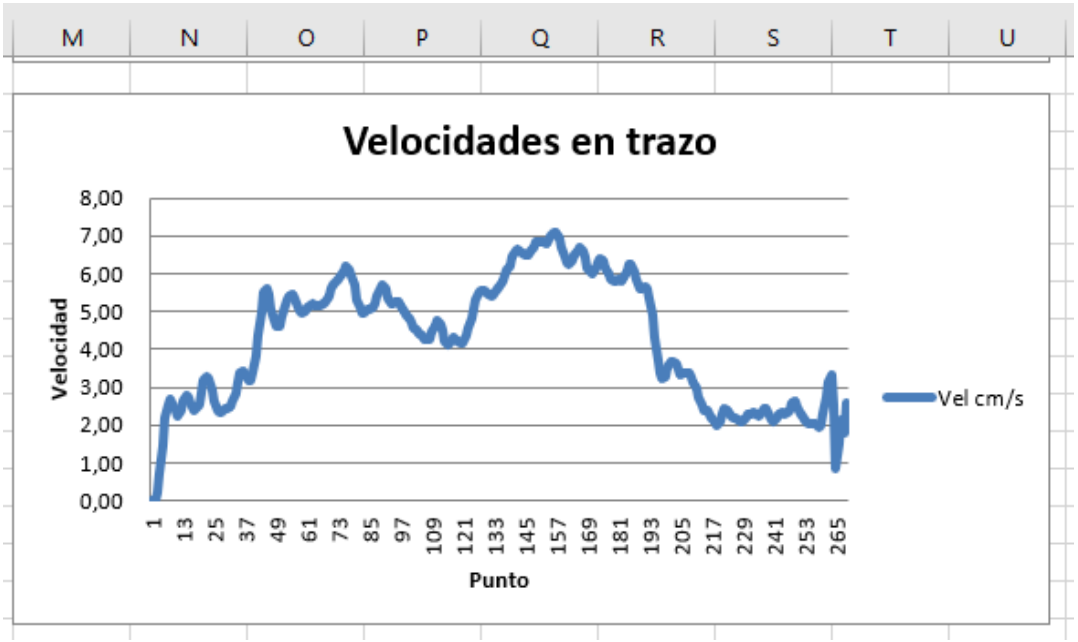


Ilustración 11 Gráfica de velocidades

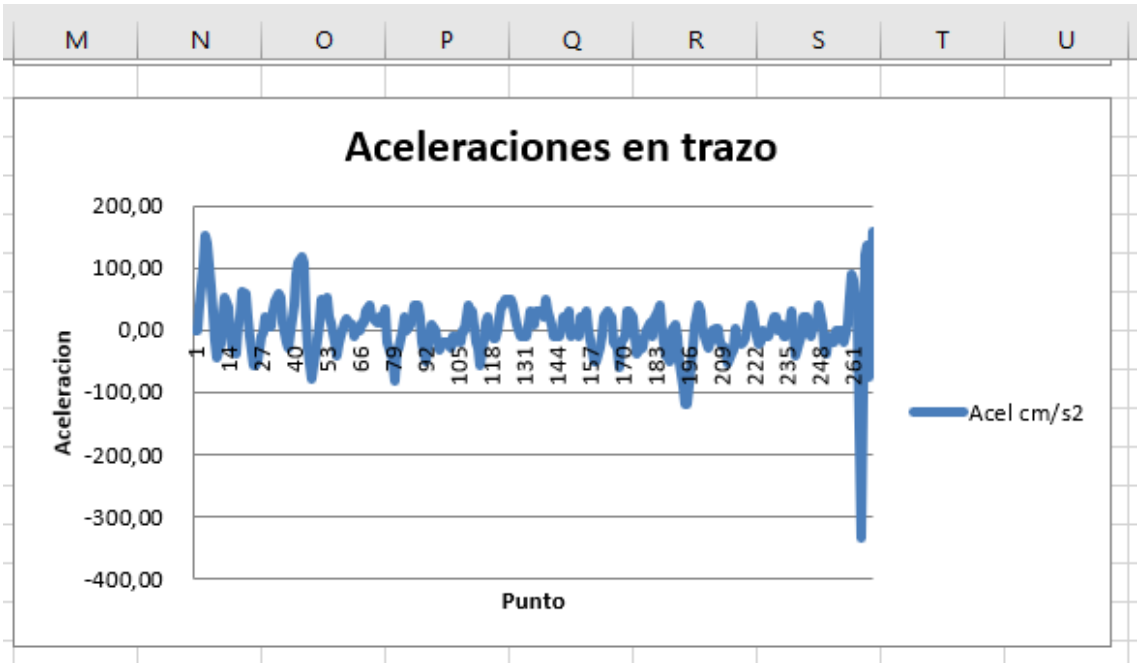


Ilustración 12 Gráfica de Aceleraciones

Descripción Experimental

Obtención de datos

Inicialmente se hace una carga del archivo correspondiente al ejercicio realizado por el niño, estos datos se analizan y se distribuyen en tres vectores, cada uno de ellos corresponderá a un dibujo diferente dentro de la misma página (debido a que cada hoja del cuaderno se compone de tres cuadrantes con tres dibujos diferentes, es necesario analizarlos de forma única y obtener valores para cada dibujo), se realizará el pasado de los datos en formato *json* a vectores mediante un bucle *for*. Estos vectores contendrán un conjunto de puntos que corresponden a las líneas dibujadas, según el valor de las x e y, se asignarán a un ejercicio u a otro, en función de en qué intervalo se encuentre la mayoría de los puntos, permitiendo que en caso de que haya más de una línea y alguna de ellas acabe sobre la línea delimitante del ejercicio o invadiendo otro espacio reservado a otro ejercicio el programa tenga total funcionalidad ante estos problemas. Además, se contará el número de trazos que ha empleado el niño para dibujar cada ejercicio, sumando cuántos *path* hay en todo el archivo.
El código con el que se realiza esto es el siguiente:

```
archivo = nombre+"-1.json"
    with open("data/"+archivo) as file:
        data = json.load(file)
```

Para cada hoja existe una función que carga el fichero *json* correspondiente a dicha hoja, pasándole el nombre (nombre del fichero) y posteriormente se agrega el "-1" correspondiente al ejercicio.
Posteriormente se llama a la función open integrada dentro de Python, busca dentro de la carpeta data (lugar donde se guardan los ficheros *json* preparados para ser evaluados tras la conversión) el fichero correspondiente y con la función load de dentro del paquete importado *json*, carga este archivo dentro de la variable data.
Se crean variables para almacenar los puntos correspondientes a cada ejercicio, el número de puntos de cada uno de ellos, las fuerzas ejercidas y la suma de todas ellas (es posible calcularlo posteriormente, pero al hacerlo a la vez que se obtienen los datos se elimina tiempo de cómputo de recorrer todos los valores dos veces).

```
for path in data[0]['paths']:
        nuevaLinea = []
        nuevaFuerza = []
        aux = mediay = numPunt = 0
        for punto in path['points']:
            nuevaLinea.append(punto)
            mediay = mediay + punto[1]
            numPunt += 1
        mediay = mediay / numPunt
        for fuerza in path['strokes']:
            nuevaFuerza.append(fuerza)
            aux += fuerza
        if mediay < 288:
            ejercicio1.append(nuevaLinea)
            numLin1 += 1
            fuerzas1.append(nuevaFuerza)
            sumFuerza1 += aux
        elif mediay> 576:
            ejercicio3.append(nuevaLinea)
            numLin3 += 1
            fuerzas3.append(nuevaFuerza)
            sumFuerza3 += aux
        else:
            ejercicio2.append(nuevaLinea)
            numLin2 += 1
            fuerzas2.append(nuevaFuerza)
            sumFuerza2 += aux
```

Dentro de la variable *data* se accede a la posición 0 para obtener todos datos, como se ha visto anteriormente, cada trazo se almacena en un *path*, por ello se hace el bucle que capture todas estas variables. Se anidan dos *for* dentro del anterior para capturar todos los puntos y todas las fuerzas, añadiéndolos a un vector auxiliar, este servirá posteriormente para guardar los puntos de cada ejercicio en su vector correspondiente.

Por cada vez que guarda valores nuevos en un ejercicio, se suma 1 al número de trazos de dicho dibujo, con las fuerzas ocurre lo mismo, solo que en vez de sumar 1, suma el valor de todas las fuerzas que ha obtenido.

Para conocer a qué ejercicio corresponde qué trazo, se tiene en cuenta las medidas de la tableta gráfica encontradas en el fichero *json* (*width*: 864, *height*: 592), para las tres primeras hojas, los ejercicios están distribuidos de la misma forma, ocupando $\frac{1}{3}$ de la página, es decir, cada ejercicio tendrá un ancho de $\frac{864}{3}$, lo que conlleva 288 píxeles de la tableta gráfica. Siguiendo el siguiente esquema para la asignación de ejercicios por píxeles:

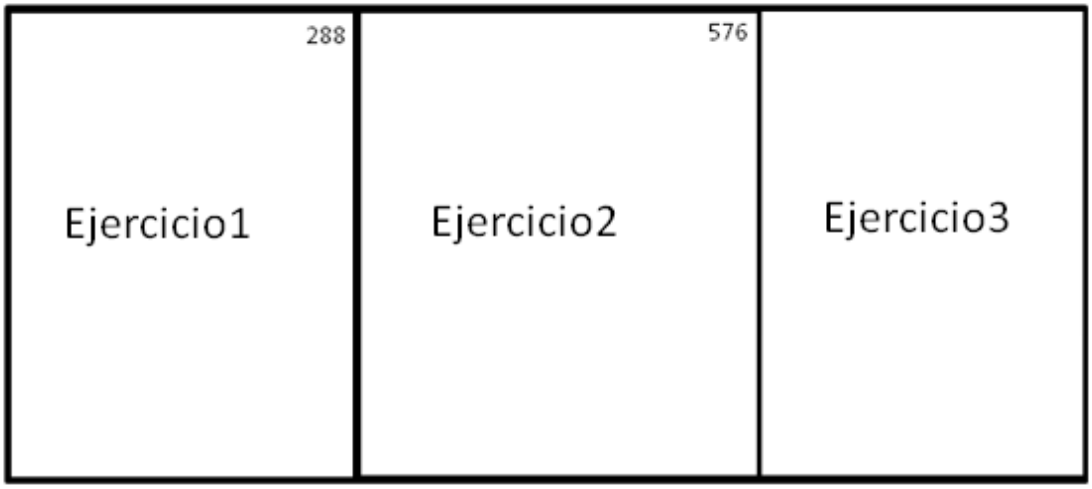


Ilustración 13 Distribución de píxeles por ejercicio

Para la última hoja esto no se aplica como se ha visto anteriormente, sino que la asignación de píxeles se realiza de la siguiente forma:

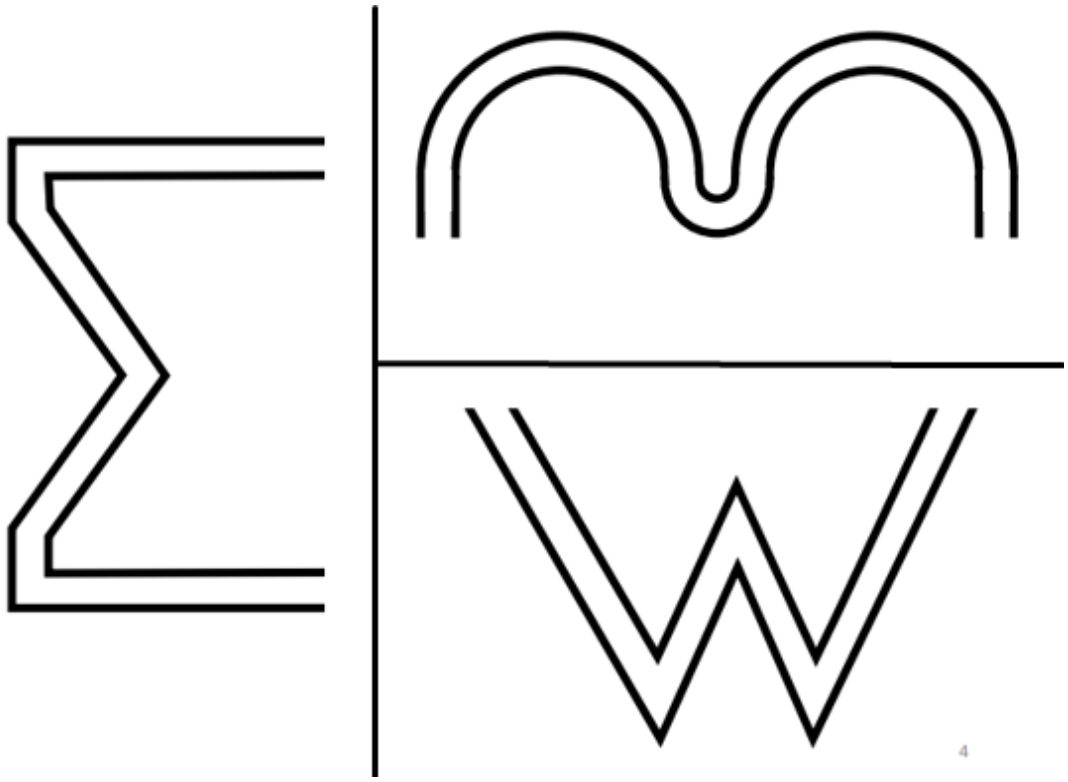


Ilustración 14 Organización de los laberintos

Al incluir las medidas de la tableta dentro del fichero json, es posible readaptar fácilmente la asignación de ejercicios por píxeles en caso de que la situación del ejercicio que se esté evaluando requiera emplear otra tableta gráfica diferente, con distintas medidas.

Cálculo punto inicial

Lo siguiente que se realiza es el cálculo de la posición de inicio de dibujado, accediendo al primer punto de las líneas del vector de cada ejercicio y comparándolo con el último punto.

```
punto1 = puntos[0]
    punto2 = puntos[longitud]

    if punto1[1]>punto2[1]:
        iniciox = "derecha"
    elif punto1[1]<punto2[1]:
        iniciox = "izquierda"
    else:
        iniciox = " "
    if punto1[0]>punto2[0]:
        inicioy = "arriba"
    elif punto1[0] < punto2[0]:
        inicioy = "abajo"
    else:
        inicioy = " "
```

El objetivo es calcular la posición relativa a la figura desde la cual se empieza a dibujar, tomando el primer punto dibujado y el último es posible conocerlo, como se ha visto anteriormente, existen tres opciones posibles para cada par de puntos (comparando las "*x*" y comparando las "*y*"), en caso de que sean iguales las "*x*" se interpreta como que es una horizontal y en caso de que sean iguales las *"y"* se tomará como una vertical. Para el resto de los casos, se interpreta, dependiendo de qué punto sea mayor, como que ha empezado por arriba, abajo, izquierda o derecha.
Es necesario aclarar que esto solo es aplicable para las rectas, en caso de querer calcularlo para figuras geométricas, se tendrá en cuenta el centroide y el primer punto, dando el resultado en formato de coordenadas (Norte, Sur, Este, Oeste, Noroeste, Suroeste, Noreste, Sureste). Posteriormente se verá cómo se calcula el centroide y la aplicación del punto inicial para las figuras.

Tiempo empleado

Viendo el tamaño de cada vector obtenemos el número de muestras:

numMuestras = len(vector)

Al dividir esto por la frecuencia se consigue el tiempo total empleado en esa línea por el niño:

Tiempo = numMuestras/0.005

De tal forma que se obtiene el tiempo que ha tomado al niño dibujar la línea, en caso de querer verlo por trazos, es necesario separar los trazos de las muestras, consiguiendo así ver qué líneas ha dibujado más rápido y en cuáles se ha tomado más tiempo para realizar.

Recta de regresión

Para calcular la recta de regresión se emplea la clase *linearRegression*, encontrada en el paquete *scikit-learn*, dentro se encuentra la función que ajusta la línea al modelo lineal. Como los puntos se guardan en formato lista para facilitar su acceso posteriormente y la fórmula encargada de calcular la recta de regresión emplea vectores, es necesario hacer una conversión de datos, separándolos en un array para x y otro para *y*. Tras esto se debe crear un objeto de tipo *LinearRegression()* y llamar a la función *fit* del mismo objeto, pasando como parámetro los valores de x e y, una vez ajustadas se llama a *predict* pasando como parámetro los *arrays* de *x* e *y* ya ajustados, obteniendo como resultado la y predicha. Además, es posible obtener el coeficiente de la recta de regresión mediante la función *coef*.

```
iguales = 0
    x_reg = np.zeros(shape=(longitud, 1))
    y_reg = np.zeros(shape=(longitud, 1))
    for i in range(longitud):
        x_reg[i] = puntos[i][0]
        y_reg[i] = puntos[i][1]
        if i > 0 and puntos[i][0] == puntos[i-1][0] and puntos[i][1] == puntos[i-1][1]:
            iguales += 0
    regressor = LinearRegression()
    regressor.fit(x_reg, y_reg)
    y_pred = regressor.predict(x_reg)
    m = regressor.coef_[0][0]
    c = regressor.intercept_[0]
```

Aprovechando que se realiza el acceso a todos los puntos almacenados se comprueba el número de veces que el niño ha dejado el lápiz parado sobre el papel, almacenando en iguales el número de puntos que coinciden (tras esto se obtiene el tiempo total parado multiplicando el número de puntos iguales por el periodo de la tableta gráfica). De tal forma, si por ejemplo el niño realiza una parada de un segundo en algún lugar del dibujo, existen 1/0.005 puntos iguales, lo que es lo mismo a 200, entonces, haciendo la inversa a esta fórmula, desde el dibujo final es posible conocer el tiempo parado, 200*0.005, dando el segundo que el niño ha parado. Estos tiempos tienen una precisión del nivel de media centésima de segundo.

Cálculo de la pendiente de la recta

En los ejercicios de rectas es posible calcular la pendiente de la línea de regresión, empleando el siguiente código:

```
def pendiente(x1, y1, x2, y2):
    if(x1!=x2):
        resultado = (y2-y1)/(x2-x1)
    else:
        resultado = 0
    return resultado

pt1x = x_reg[1]
pt1y= y_reg[1]
pt2x = x_reg[longitud-2]
pt2y = y_reg[longitud-2]
pendienteIdeal = pendiente(pt1x, pt1y, pt2x, pt2y)
```

Este código coge el punto *x* y el punto *y* de la recta de regresión al principio y al final. Con esto se llama a la función pendiente que se define anteriormente.
En la función se comprueba el riesgo de error por dividir entre 0 (recordando la fórmula de la pendiente, en el denominador se calcula la diferencia entre las *x*, si este valor es el mismo se producirá una división entre 0, saltando una excepción). En caso de que sean distintos los valores de *x1* y *x2* se calcula el resultado restando las *y*, restando las *x* y haciendo la división entre ambos resultados obtenidos. El resultado se devuelve almacenándose en *pendienteIdeal*, pudiendo ser usado posteriormente por el programa para crear las tablas y mostrar el resultado en el Excel asociado al ejercicio que se está evaluando, en caso de querer mostrar menos decimales, es posible emplear el comando *round* posteriormente a la llamada de la función pendiente para así adecuar al formato especificado por los profesionales que emplearán la aplicación.

Cálculo ángulo de la recta

Una vez se obtenga la pendiente, es posible conocer el ángulo de la recta que ha sido dibujada (a partir de la regresión lineal), el código es el siguiente:

```
if pendienteIdeal!=0:
    anguloRecta = round(math.degrees(math.atan(pendienteIdeal)),2)
elif pt1x==pt2x:
    anguloRecta = 0
else:
    anguloRecta = 90
```

En caso de que la pendiente sea 0, significa que es una recta horizontal o vertical, comprobando si las *x* o las *y* son iguales se obtiene el ángulo para estas dos opciones.
Para el resto de los casos (cuando la pendiente ideal es distinta de 0), se calcula la arcotangente de la pendiente (*math.atan*), se hace la conversión de radianes a grados (*math.degrees*) y por último se redondea a dos decimales (round).

Cálculo de velocidad, aceleración y distancia por tiempo

Conociendo el número de puntos es posible obtener cuatro segmentos de misma longitud, dividiendo el número de puntos entre cuatro y multiplicándolo por el periodo se obtiene el tiempo que se emplea a la hora de dibujar ese segmento.

A partir de estos segmentos se calcula la velocidad, aceleración y distancia con el fin de saber cuándo va más deprisa y cuándo más despacio, la relación entre la velocidad y la presión y la aceleración media de cada uno de estos cuadrantes.

En caso de que haya un número impar de puntos, es necesario ajustar este valor a un múltiplo de dos, es decir, eliminar el último punto a la hora de calcular los puntos que compondrán todos estos segmentos, para así evitar problemas de fallo de rangos por uso de decimales en el acceso. Por ejemplo, si hay 445 puntos, cada cuadrante irá de 111.25 en 111.25, por ello, al eliminar un punto se soluciona el problema de que los vectores traten de acceder en coma flotante y no en entero, 445 – 1 = 444, por lo tanto, de 111 en 111 irán cada uno de los segmentos correspondientes al dibujo realizado por el alumno.

El código encargado de calcular las distancias es el siguiente:

```
primero = round(math.sqrt((puntos[0][0] - puntos[intermedios-1][0])**2 +
(puntos[0][1] - puntos[intermedios-1][1])**2),3)

segundo = round(math.sqrt((puntos[intermedios][0] - puntos[intermedios*2-
1][0])**2 + (puntos[intermedios][1] - puntos[intermedios*2-1][1])**2),3)

tercero = round(math.sqrt((puntos[intermedios*2][0] - puntos[intermedios*3-
1][0])**2 + (puntos[intermedios*2][1] - puntos[intermedios*3-1][1])**2),3)

cuarto = round(math.sqrt((puntos[intermedios*3][0] - puntos[intermedios*4-1][0])**2
+ (puntos[intermedios*3][1] - puntos[intermedios*4-1][1])**2),3)

distanciatot = (primero+segundo+tercero+cuarto)/40
```

En caso de las velocidades:

```
vel1 = (primero/40)/(intermedios*0.005)
vel2 = (segundo/40)/(intermedios*0.005)
vel3 = (tercero/40)/(intermedios*0.005)
vel4 = (cuarto/40)/(intermedios*0.005)
velMedia = (vel1+vel2+vel3+vel4)/4
```

Por último, para las aceleraciones:

```
acel1 = (vel1-0)/(intermedios*0.005)
acel2 = (vel2-vel1)/(intermedios*0.005)
acel3 = (vel3-vel2)/(intermedios*0.005)
acel4 =(vel4-vel3)/(intermedios*0.005)
acelMedia = (acel1 + acel2 + acel3 + acel4)/4
```

Cálculo de área y perímetro

Para las figuras geométricas planas (circulo, cuadrado y triángulo) es posible calcular el área y el perímetro a partir de la lista de puntos. Se emplean tres funciones para ello, *segments*, *perimeter* y *área_fig*, estas son las encargadas de agrupar los datos de los segmentos, calcular el perímetro y el área, el código es:

```
def segments(poly):
        """A sequence of (x,y) numeric coordinates pairs """
        return zip(poly, poly[1:] + [poly[0]])

def perimeter(poly):
    """A sequence of (x,y) numeric coordinates pairs """
    return abs(sum(math.hypot(x0-x1,y0-y1) for ((x0, y0), (x1, y1)) in segments(poly)))

def area_fig(poly):
    """A sequence of (x,y) numeric coordinates pairs """
    return 0.5 * abs(sum(x0*y1 - x1*y0 for ((x0, y0), (x1, y1)) in segments(poly)))
```

- *Segments* agrupa las coordenadas en forma de pares, para poder leerlas posteriormente en el perímetro y el área, obteniendo una lista de valores *x* e *y* correspondientes a la figura que recibe inicialmente, ordenando correctamente todos los puntos para su posterior uso en las funciones. Mapeando los índices que recibe para un acceso más ordenado y cómodo.

- *Perimeter* devuelve el valor absoluto de la hipotenusa de la diferencia de los puntos x e y consecutivos en el array de segmentos, lo cual corresponde al perímetro de la figura. Emplea un acceso secuencial a cada uno de los valores mediante la sentencia *for*, empleando el nombrado previo en la función *segments* para facilitar la ejecución permitiendo el acceso múltiple a los datos, liberando computación del bucle, el cual añadiría más líneas de código que deben ser compiladas y ejecutadas, empeorando los tiempos de ejecución de la aplicación final.

- *Area_fig* devuelve la mitad de la suma de la multiplicación de x e y para puntos consecutivos del array de segmentos, correspondiéndose con el área de la figura. Al igual que *perimeter*, accede de forma secuencial a los valores previamente mapeados, reduciendo tiempo de cómputo y facilitando al psicólogo el evaluar a más personas por tiempo.

Todas estas funciones reciben como parámetro "*poly*", que se corresponde con la lista de puntos de la figura dibujada por el alumno. Como se emplea el mismo fichero para evaluar tanto las cruces como las figuras geométricas, se diferencia si se aplican estas funciones mediante un booleano en la llamada de la función (explicado posteriormente en el diagrama del programa).

Cálculo circularidad

Para ver cómo de perfectos son los círculos dibujados por los alumnos se evalúa la circularidad basándose en la formula descrita anteriormente y haciendo uso de las funciones de perímetro y área. Basándose en los momentos de Hu el resultado obtenido va de 0 a 1, siendo 1 la circunferencia perfecta.
Para obtener el valor exacto de pi en vez de emplear uno acortado (3.1416) se emplea el paquete math y dentro del mismo el valor pi, el cuál coincide con el número irracional.
El perímetro y el área de la figura se consigue mediante las funciones *perimeter* y *área_fig* explicadas anteriormente, con ello se multiplica el área por cuatro pi y se divide entre el perímetro al cuadrado, obteniendo el valor de Hu correspondiente.
El código empleado es:

```
perimetro = perimeter(puntos)
area = area_fig(puntos)
F=(4*math.pi*area)/(perimetro**2)
```

Cálculo centroide

En las figuras geométricas planas es posible calcular el centroide, lo cual servirá para conocer el punto de inicio de la figura y cómo de centrados se encuentran todos los puntos del perímetro. Se calcula sumando los valores de los puntos *x* e *y* y dividiéndolo posteriormente entre el número de puntos, con esto se obtienen las coordenadas de *x* e *y* correspondientes. Posteriormente en la lógica de selección de orientación inicial de las figuras se empleará. Además, sirve a la hora de estudiar los valores de las tablas para hacerse una idea de la posición relativa de la figura con los cuadrados marcados para el dibujo del ejercicio por parte del niño.

La forma de calcularlo es mediante el siguiente código:

```
x = [p[0] for p in puntos]
y = [p[1] for p in puntos]
centroide = (sum(x)/len(puntos), sum(y)/len(puntos))
```

Presiones

```
longitud = len(fuerzas)
mediaFuer = round(sumFuerza/(longitud),3)
cuartos = round(longitud/4)
for i in [0,1,2,3]:
    sumaFParcial = 0
    num = 0
    for j in range(cuartos):
        sumaFParcial += fuerzas[j+cuartos*i]
        num += 1
     sumaFParcial = sumaFParcial / num
     presiones.append(sumaFParcial)
```

Para obtener la lógica de las presiones se emplea un fichero exclusivo para ello, este recibe la suma de fuerzas, la lista con las presiones y atributos relacionados con el Excel.
En *longitud* se almacena el número de fuerzas (puntos), tras esto se calcula la media de las fuerzas redondeando a tres decimales, para calcularlo se coge la suma de todas las fuerzas y se divide entre el número de puntos o fuerzas (siendo el mismo).
A continuación, se calcula en cuartos el número de puntos que va a contener cada uno de los cuatro segmentos, teniendo en cuenta el número de elementos que existen, para evitar números impares y problemas de vectores por intentar acceder mediante valores en coma flotante a cada uno de los segmentos. Para cada uno de ellos se calcula la suma de todas sus fuerzas y se divide entre el número de fuerzas de dicho segmento, obteniendo así la media de fuerza ejercida al principio, mitad y final del dibujo, facilitando el estudio de en qué momentos se ha ejercido más presión y en cuales menos, derivando así en la obtención de cuándo ha querido tener más precisión (apretando menos) y su análogo a la hora de apretar más en el trazo.

Creación fichero Excel

Como cada prueba VMI se almacena en un Excel distinto es necesario crear un programa que cree estos archivos para facilitar el uso de la aplicación al usuario, para ello se emplea el paquete *Openpyxl*.
Existe la posibilidad de realizar una prueba a un mismo niño dos veces en el mismo día, por ello hay que añadir un identificador final para los Excel repetidos.
Para identificar el Excel con el alumno al que se le está realizando el estudio, se escribe el nombre del fichero *json* que corresponde a sus datos, sin incluir el número del ejercicio:

```
contenido = os.listdir('excel/')
nombre = "child12-20.04.21"
persona = nombre
while persona+'.xlsx' in contenido:
    persona+="A"
wb = openpyxl.Workbook()
wb.save("excel/"+persona+'.xlsx')
```

Importando el paquete os podremos acceder a listar directorios, para conocer si existe un fichero con el nombre del ejercicio que se ha escrito, en caso de que exista se añade un "*A*" al nombre, obteniendo así la segunda versión del mismo día.9

Nombre corresponde con el niño al que se está evaluando junto con la fecha en la que se ha realizado la prueba. Esto está mejor explicado en el manual de usuario, junto con cómo crear los archivos necesarios para la ejecución de este código.

Finalmente se crea un nuevo archivo y se guarda dentro de la carpeta "*excel*" del programa, con el nombre obtenido anteriormente.

Por ejemplo, para el niño 12, según el código, se obtendrá el fichero Excel *child12-20.04.21.xlsx*, en caso de que ya exista un archivo con ese nombre, significando que ese alumno ya realizó una prueba ese mismo día, se guardará como Excel *child12-20.04.21A.xlsx*. Todo esto dentro de la carpeta Excel de la aplicación, con el fin de obtener el resultado de forma más ordenada posible, según el explorador de archivos de Windows 10, a la hora de ordenar los archivos por nombre lo realizará de forma alfabética, dejando juntos todos los Excel correspondientes a cada niño, pudiendo abrir fácilmente todos los que se deseen por ejemplo para ver la evolución de esa persona.

Inclusión de puntos, presiones, velocidades y aceleraciones

A la hora de cargar los puntos dentro de la aplicación (Obtención de datos), es el mejor momento para coger los valores de presión, *x* e *y* de cada punto, con el fin de evitar volver a recorrer todos los valores más de una vez, reduciendo así el tiempo de ejecución del programa.

Se almacenan las distancias, velocidades y aceleraciones en un *array* por cada elemento (los cuales son creados rellenándose con el mismo número de ceros que de elementos vayan a contener posteriormente), empleando las fórmulas de aceleración y velocidad se calculan las mismas para cada punto en relación con el punto anterior.

Estos cálculos son realizados para cada par de puntos, relacionando el anterior con el actual, a excepción del primer valor, ya que eso daría una excepción por acceso con número negativo al array. Así se ve la evolución de aceleraciones, presiones y velocidades a lo largo de cada punto ejercido, muy útil para ampliar exactamente dentro de cada cuadrante, en qué parte se ha empezado a ir más deprisa o se ha disminuido la presión por ejemplo.

El código que obtiene estos *arrays* y los rellena es el siguiente:

```
dist = np.array(np.zeros(numPunt))
vel  = np.array(np.zeros(numPunt))
acel = np.array(np.zeros(numPunt))
for j in range(1, numPunt):
   dist[j] =  ((pos[j][0]-pos[j-1][0])**2 + (pos[j][1]-pos[j-1][1])**2)** 0.5

vel  = (dist / 40 ) / 0.005

for j in range(1, numPunt):
   acel[j] =  (vel[j] - vel[j-1]) / 0.005
suma = np.c_[pos, pres, dist, vel, acel]
```

Una vez se tengan todos los datos almacenados es necesario crear las hojas correspondientes a cada ejercicio, como por cada fichero *json* existen tres dibujos, se realiza la carga de datos de tres en tres, por lo que se crean tres hojas Excel.

```
ws1 = wb.create_sheet(nombre+"-1")
ws2 = wb.create_sheet(nombre+"-2")
ws3 = wb.create_sheet(nombre+"-3")
```

Al estar creadas, ya se puede comenzar a cargar los datos, se emplea la lógica de elegir a qué ejercicio corresponde en función de la media de los valores de x e y, una vez determinado se ejecuta el siguiente código:

```
ws1.cell(row=filax1, column=1).value = "Trazo "+str(numLin1)
ws1.cell(row=filax1, column=2).value = "( " + str(numPunt) + " muestras)"
filax1 += 1
ws1.cell(row=filax1, column=2).value = "X"
ws1.cell(row=filax1, column=3).value = "Y"
ws1.cell(row=filax1, column=4).value = "Pres"
ws1.cell(row=filax1, column=5).value = "Dist (pts)"
ws1.cell(row=filax1, column=6).value = "Vel cm/s"
ws1.cell(row=filax1, column=7).value = "Acel cm/s2"
muestra = 0
for fila, datos in enumerate(suma):
   filax1 += 1
   muestra += 1
   ws1.cell(row=filax1, column=colx1 - 1).value = muestra
```

```
            for col, dato in enumerate(datos):
                # print (fila, col, dato)
                ws1.cell(row=filax1, column=col+colx1).value = dato
                ws1.cell(row=filax1, column=col+colx1).number_format = formato
        values = Reference(ws1,min_col = 4, max_col=4,min_row=filax1-numPunt,
max_row=filax1)
        grafico.add_data(values, titles_from_data=True)
        grafico.title = "Fuerzas en trazo "
        grafico.style = 10
        grafico.x_axis.title = "Punto"
        grafico.y_axis.title = "Fuerza"

        valuesV = Reference(ws1,min_col = 6, max_col=6,min_row=filax1-numPunt,
max_row=filax1)
        grafVel.add_data(valuesV, titles_from_data=True)
        grafVel.title = "Velocidades en trazo "
        grafVel.style = 10
        grafVel.x_axis.title = "Punto"
        grafVel.y_axis.title = "Velocidad"

        valuesA = Reference(ws1,min_col = 7, max_col=7,min_row=filax1-numPunt,
max_row=filax1)
        grafAcel.add_data(valuesA, titles_from_data=True)
        grafAcel.title = "Aceleraciones en trazo "
        grafAcel.style = 10
        grafAcel.x_axis.title = "Punto"
        grafAcel.y_axis.title = "Aceleracion"
        filax1 += 1
        ws1.cell(row=filax1, column=1).value = "TOTAL"

        ws1.cell(row=filax1,
        column=6).value = dist.sum()/40
        ws1.cell(row=filax1, column=6).number_format = formato
        ws1.cell(row=filax1, column=7).value = np.mean(vel)
        ws1.cell(row=filax1, column=7).number_format = formato
        filax1 += 2
        celda = "M"+str(filax1-numPunt)
        ws1.add_chart(grafico,celda)

        celda = "M"+str(filax1-numPunt+15)
        ws1.add_chart(grafVel,celda)

        celda = "M"+str(filax1-numPunt+30)
        ws1.add_chart(grafAcel,celda)
```

Las variables *grafVel*, *grafico* y *grafAcel* son de tipo *LineChart(),* han de crearse tres distintas ya que no es posible reutilizar una variable gráfico para dibujar más de uno.
Para cada trazo que se detecta en la obtención de datos, se añade un conjunto de puntos, en función del ejercicio al que corresponda, por lo que existe este código para *ws2* y *ws3* (diferentes hojas para cada dibujo).
En la primera fila se incluye el trazo que se está dibujando (Trazo 1, Trazo 2 …), a continuación, se incluyen los títulos de cada columna (*X, Y, Pres, Dist* (pts), *Vel cm/s* y *Acel cm/s2*) y tras esto se incluyen todos los valores, autoincrementando la fila, consiguiendo que cada valor insertado corresponda con la columna indicada.
Para la selección de la columna correspondiente al gráfico se emplea "Reference", donde se indica el fichero en el que se va a escribir (hoja en este caso), las columnas que se desean leer y las filas donde se encuentran los datos.
Tras ello se cargan los datos en el gráfico correspondiente con "add_data", se le establece un título, un estilo para que indique el tamaño, se insertan valores a los ejes x e y.
A continuación, se inserta la suma de todos los valores añadidos anteriormente junto a un TOTAL. Se selecciona en qué celda se desea insertar la gráfica y tras ello se emplea la función "add_chart", donde se especifica la hoja en la que se añade, la celda y a qué gráfico se refiere.
En los ficheros "recta.py", "circulo.py" y "figura.py" se realiza la evaluación de los diferentes dibujos, todos los datos recogidos son almacenados en diferentes variables, todo esto debe aparecer en el Excel resultante y en la hoja correspondiente, estas funciones reciben como parámetro el archivo "ws" correspondiente a la hoja donde deben escribir, "wb" como el archivo Excel creado anteriormente, el número del ejercicio y el nombre del archivo Excel.
Al tener la hoja (ws) es posible conocer el número de columnas escritas (ws.max_column), con ello se accederá a la tercera columna disponible tras la última (con el fin de dejar espacio suficiente entre las tablas) y se inserta en ella los nombres de los valores que van a ser añadidos y en la siguiente los valores correspondientes, como el objetivo es realizar una tabla, cada valor nuevo debe sumar uno a la fila. Para acceder a cada celda se emplea:

ws.cell(row=..., column=...).value = valor

El código empleado para la función recta que carga los datos es el siguiente:

```
    ws.cell(row = fila +6, column= columna).value = "distancia en el cuarto 1 (cm)"
    ws.cell(row = fila +7, column= columna).value = "distancia en el cuarto 2 (cm)"
    ws.cell(row = fila +8, column= columna).value  = "distancia en el cuarto 3 (cm)"
    ws.cell(row = fila +9, column= columna).value  = "distancia en el cuarto 4 (cm)"
    ws.cell(row = fila +6, column= columna+1).value = primero/40
    ws.cell(row = fila +7, column= columna+1).value = segundo/40
    ws.cell(row = fila +8, column= columna+1).value = tercero/40
    ws.cell(row = fila +9, column= columna+1).value= cuarto/40
    ws.cell(row = fila +10, column= columna).value = "distancia total (cm)"
    ws.cell(row = fila +10, column= columna+1).value= distanciatot
    ws.cell(row = fila +11, column= columna).value= "ha ido más rápido"
    ws.cell(row = fila +12, column= columna).value = "ha ido más lento"
    ws.cell(row = fila +13, column= columna).value = "velocidad media"
    ws.cell(row = fila +13, column= columna+1).value = velMedia
    ws.cell(row = fila +14, column= columna).value = "velocidad en el cuarto 1 (cm/s)"
    ws.cell(row = fila +14, column= columna+1).value= vel1
    ws.cell(row = fila +15, column= columna).value = "velocidad en el cuarto 2 (cm/s)"
    ws.cell(row = fila +15, column= columna+1).value = vel2
    ws.cell(row = fila +16, column= columna).value = "velocidad en el cuarto 3 (cm/s)"
    ws.cell(row = fila +16, column= columna+1).value = vel3
    ws.cell(row = fila +17, column= columna).value = "velocidad en el cuarto 4 (cm/s)"
    ws.cell(row = fila +17, column= columna+1).value = vel4
    ws.cell(row = fila +18, column= columna).value = "aceleracion media"
    ws.cell(row = fila +18, column= columna+1).value= acelMedia
    ws.cell(row = fila +19, column= columna).value = "aceleracion en el cuarto 1
(cm/s)"
    ws.cell(row = fila +19, column= columna+1).value= acel1
    ws.cell(row = fila +20, column= columna).value = "aceleracion en el cuarto 2
(cm/s)"
    ws.cell(row = fila +20, column= columna+1).value = acel2
    ws.cell(row = fila +21, column= columna).value = "aceleracion en el cuarto 3
(cm/s)"
    ws.cell(row = fila +21, column= columna+1).value = acel3
    ws.cell(row = fila +22, column= columna).value = "aceleracion en el cuarto 4
(cm/s)"
    ws.cell(row = fila +22, column= columna+1).value = acel4

    ws.cell(row = fila +23, column= columna).value = "punto inicio"
    ws.cell(row = fila +23, column= columna+1).value= iniciox + " " +inicioy
    ws.cell(row = fila +24, column= columna).value = "Pendiente Recta"
    ws.cell(row = fila +24, column= columna+1).value = pendienteIdeal[0][0]
```

```
if primero > segundo and primero > tercero and primero > cuarto:
    ws.cell(row = fila + 11, column= columna+1).value = 1
elif segundo > primero and segundo > tercero and segundo > cuarto:
    ws.cell(row = fila + 11, column= columna+1).value = 2
elif tercero > segundo and tercero > primero and tercero > cuarto:
    ws.cell(row = fila + 11, column= columna+1).value= 3
elif cuarto > segundo and cuarto > tercero and cuarto > primero:
    ws.cell(row = fila + 11, column= columna+1).value = 4
elif primero == segundo and segundo == tercero and tercero == cuarto:
    ws.cell(row = fila + 11, column= columna+1).value = "velocidad constante"
if primero < segundo and primero < tercero and primero < cuarto:
    ws.cell(row = fila + 12, column= columna+1).value = 1
elif segundo < primero and segundo < tercero and segundo < cuarto:
    ws.cell(row = fila + 12, column= columna+1).value= 2
elif tercero < segundo and tercero < primero and tercero < cuarto:
    ws.cell(row = fila + 12, column= columna+1).value= 3
elif cuarto < segundo and cuarto < tercero and cuarto < primero:
    ws.cell(row = fila + 12, column= columna+1).value = 4
ws.cell(row =  fila + 1, column= columna).value= "ejercicio"
ws.cell(row =  fila + 1, column= columna+1).value= exer
ws.cell(row =  fila + 2, column= columna).value= "Angulo recta"
ws.cell(row = fila + 2, column= columna+1).value = anguloRecta
ws.cell(row = fila + 3, column= columna).value = "Num Puntos"
ws.cell(row = fila +3, column= columna+1).value= longitud+1
ws.cell(row = fila +4, column= columna).value = "Numero trazos"
ws.cell(row = fila +4, column= columna+1).value= numLin
ws.cell(row = fila +5, column= columna).value= "Tiempo (s)"
ws.cell(row = fila +5, column= columna+1).value= tiempo
```

Este código específicamente inserta los datos de la hoja *recta.py*, cada una de las hojas diferentes insertan unos valores u otros, ya que como se ha visto en la descripción teórica, en algunos dibujos no tiene sentido calcular la pendiente de la recta, así como en otros tampoco lo tiene ver el centroide o el área.

Tras esto es necesario ajustar el formato de las cuadriculas para poder ver los datos correctamente sin tener que aumentarlo manualmente, además de incluir un color de fondo y marcar los bordes para facilitar la lectura y hacer más cómoda la relación de valores con nombres.

Se consigue utilizando el código que sigue:

```
thin_border = Border(left=Side(style='thin'),
                     right=Side(style='thin'),
                     top=Side(style='thin'),
                     bottom=Side(style='thin'))

whiteFill = PatternFill(fgColor="00FFFFCC",fill_type="solid")

    for i in range(fila+1,fila+30):
        ws.cell(i, columna).fill = whiteFill
        ws.cell(i, columna+1).fill = whiteFill
        ws.cell(i, columna).border = thin_border
        ws.cell(i, columna+1).border = thin_border

    ws.column_dimensions["J"].width = 35
    ws.column_dimensions["K"].width = 17
```

Las variables *thin_border* y *whiteFill* corresponden con el borde fino que se va a añadir y el color de fondo, siendo un borde sólido negro y un relleno amarillo clarito respectivamente.
El bucle *for* que recorre todas las celdas de la tabla va de fila+1 (fila en la que empieza, ya que fila comienza valiendo 2 y se escribe el primer valor en la fila 3) a fila+30 (por las cinco celdas que añade la función de presiones a la tabla más las 24 filas que escribe el programa, siendo de un valor fijo para el programa de rectas, para los demás los valores varían ya que no todos los ficheros añaden el mismo número de valores).
Tras esto se especifican las dimensiones de la columna J y K, en especial el ancho, correspondientes a la tabla, para evitar volver a recorrer todas las celdas, lo cuál llevaría más tiempo de cómputo y peores tiempos de ejecución, se coloca el número máximo obtenido tras ver el valor medio de los datos insertados, 35 y 17. Con esto se consigue una mejor organización de los valores y una muestra más cómoda para la vista evitando que en las columnas se escriban dos líneas o que sea necesario alargar la columna para ver el contenido de dicha celda, adaptándolas a los estándares de decimales y longitud de valores que se insertan a la hora de analizar el dibujo realizado por el alumno.
Como se ha especificado anteriormente, en cada hoja existirán los datos correspondientes a cada dibujo, en la imagen a continuación se puede ver cómo sería una de estas hojas para el niño 12 (*child12*), observando que, al haber dos trazos, se crean gráficas específicas para cada uno de ellos, mientras que la tabla de resultados es única por ejercicio, evaluando todos los trazos en conjunto.
A continuación se muestra en dos imágenes cómo queda una hoja para uno de los ejercicios, viendo cómo se analizan trazos diferentes y el orden de las gráficas al lado de cada trazo, facilitando así la comprensión por parte del profesional de los datos insertados.

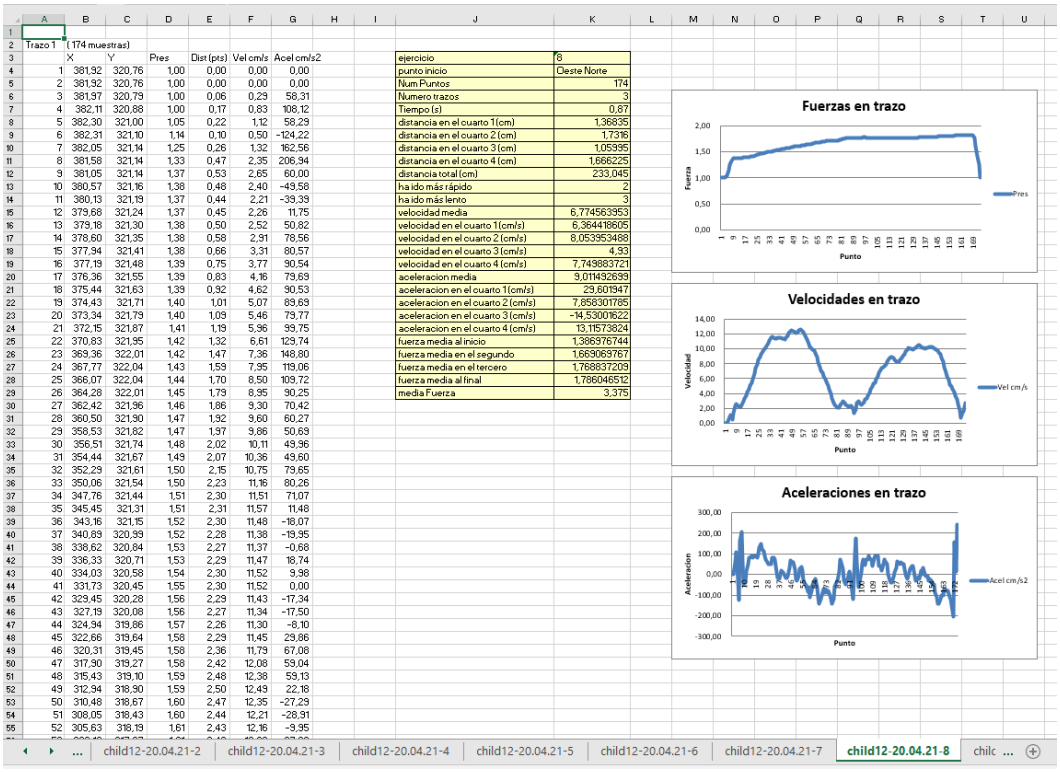


Ilustración 15 Hoja Excel para un ejercicio

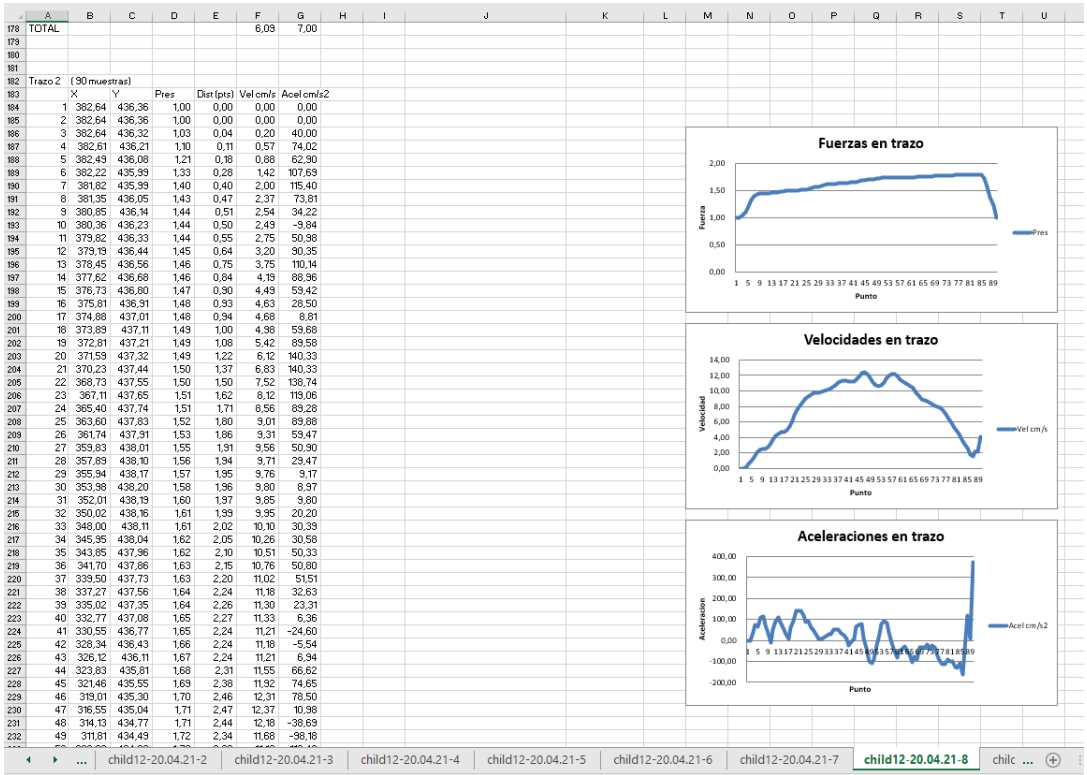


Ilustración 16 Hoja Excel mostrando segundo trazo

Resumen de los dibujos

Al ejecutar "*estudio.py*", además de llamar al resto de funciones de la aplicación, se crea el fichero Excel donde se guardarán todos los datos y posteriormente, se cargan en la hoja principal el resumen de todas las tablas de los dibujos mediante el código mostrado a continuación:

```
Hojas =wb.get_sheet_names()
numHojas = len(Hojas)
filaDest = 1
colDest = 1
fecha = Hojas[1].find('-') +1
destino.cell(row = filaDest, column = colDest).value = Hojas[1][fecha:]
filaDest+=2
print("Exportando datos a Excel")
for i in range(1,13):
    ws1 = wb.get_sheet_by_name(Hojas[i])
    for j in range(3,32):
        c = ws1.cell(row = j, column = 10)
        d = ws1.cell(row = j, column = 11)
        destino.cell(row = filaDest, column = colDest).value = c.value
        destino.cell(row = filaDest, column = colDest).fill = PatternFill("solid",
fgColor=COLOR_INDEX[26])
        destino.cell(row = filaDest, column = colDest).border = thin_border
        destino.cell(row = filaDest, column = colDest+1).value = d.value
        destino.cell(row = filaDest, column = colDest+1).fill = PatternFill("solid",
fgColor=COLOR_INDEX[26])
        destino.cell(row = filaDest, column = colDest+1).border = thin_border
        filaDest +=1
    if i%3==0:
        colDest = 1
        filaDest +=2
    else:
        colDest += 3
        filaDest = filaDest+2-31

dims = {}
for row in destino.rows:
    for cell in row:
        if cell.value:
            dims[cell.column_letter] = max((dims.get(cell.column_letter, 0),
len(str(cell.value))))
```

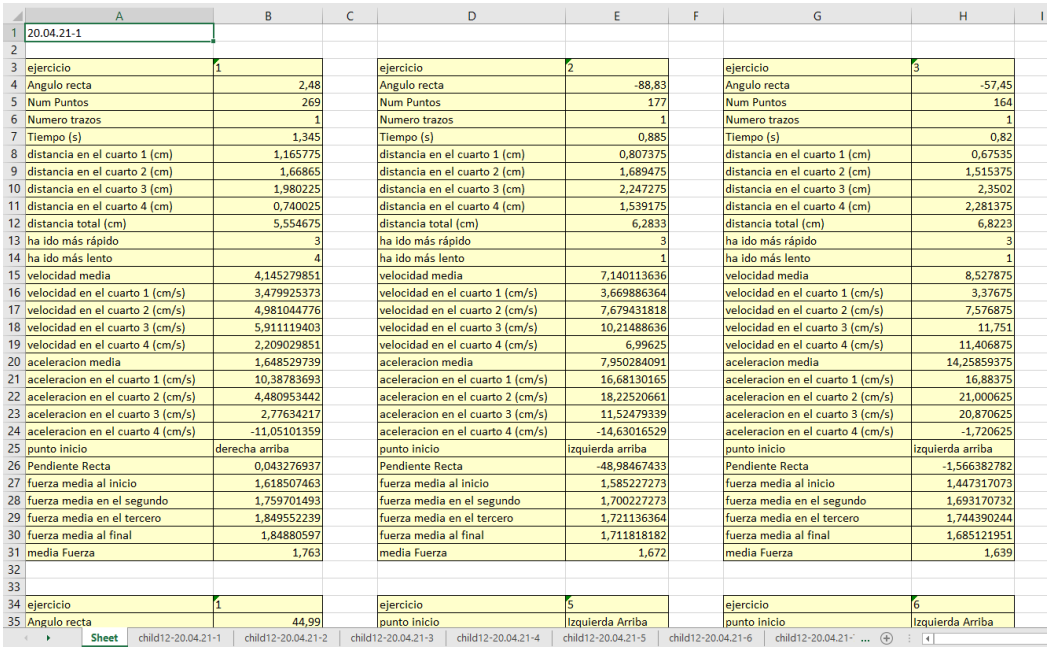
Siendo *Hojas* una lista con todos los nombres de las diferentes hojas que componen el Excel, *numHojas* el número de hojas del que se compone dicha lista, es decir, la longitud, *filaDest* y *colDest* la fila y columna respectivamente donde se realizará la escritura en la primera hoja del archivo (dichas variables son incrementadas en función de qué tabla se incluya para mostrar el máximo número por fila de la forma más cómoda a la vista).
Obteniendo el nombre de la segunda hoja se mira la fecha en la que se realizó la prueba, para así insertarla al principio del documento.
Tras esto se accede a cada una de las hojas, se obtienen los valores de las columnas correspondientes a las tablas (J y K) y se insertan uno tras otro.
El objetivo es tener 3 tablas por fila, con una separación de tres columnas entre ellas, por ello cuando el número que recorre el bucle es divisible entre tres (indicando que ya se han insertado tres tablas en esa fila), la columna vuelve a tener el valor inicial y a la fila se le suman dos (en ese momento la fila vale la posición final en la que ha escrito la tabla, haciendo que se respeten las distancias entre tablas de filas, con esa distancia de dos se facilita la comprensión del resultado obtenido). En caso de que no tenga un valor divisible entre tres, se añadirá la tabla a la derecha de la anterior, debiendo restar el correspondiente número de filas para que estén alineadas, volviendo a la posición inicial o en caso de que se haya escrito más de una fila, volver al punto de inicio de dicha fila, por ello al valor correspondiente a las filas se le suma dos (de la distancia entre una fila y la siguiente) y se le resta 31 (número de posiciones de valores que se insertan de media), y sumando tres a la columna para dejar el espacio correspondiente entre una y la siguiente. Cabe destacar que con el fin de que estén las tablas alineadas en filas se inserta un número fijo de valores, siendo el máximo de los valores que contienen las hojas, aunque deja espacios en blanco rellenos en algunas tablas, es la solución más óptima, ya que al escribir debajo de las tablas en las hojas correspondientes a cada ejercicio, si se busca el número máximo de celdas escritas por columna da un número inexacto.

```
dims = {}
for row in destino.rows:
    for cell in row:
        if cell.value:
            dims[cell.column_letter] = max((dims.get(cell.column_letter, 0),
len(str(cell.value))))
for col, value in dims.items():
    destino.column_dimensions[col].width = value-2

wb.save("excel/"+persona+'.xlsx')
```

Con el código anterior correspondiente al bucle que accede a todas las columnas escritas, especificando el tamaño máximo de los datos escritos, estableciendo el mismo como ancho de cada columna. Optimizando así el espacio empleado para cada tabla y mostrando los valores de forma que ocupen solo una línea y se vean a simple vista, sin tener que modificar la longitud de las columnas de manera manual. El funcionamiento está basado en obtener los valores de todas las columnas dentro de la hoja principal, una vez ya rellena, con ello se obtiene un array de elementos donde se tienen los valores máximos de cada columna, asociando posteriormente cada una de estas longitudes al ancho de la columna correspondiente, obteniendo un formato más cómodo de leer e interpretar. Posteriormente se realiza el salvado de datos del archivo Excel dentro de la carpeta "*excel*" del programa.

El resultado de la hoja principal es el siguiente:



	A	B	C	D	E	F	G	H	I
1	20.04.21-1								
2									
3	ejercicio	1		ejercicio	2		ejercicio	3	
4	Angulo recta	2,48		Angulo recta	-88,83		Angulo recta	-57,45	
5	Num Puntos	269		Num Puntos	177		Num Puntos	164	
6	Numero trazos	1		Numero trazos	1		Numero trazos	1	
7	Tiempo (s)	1,345		Tiempo (s)	0,885		Tiempo (s)	0,82	
8	distancia en el cuarto 1 (cm)	1,165775		distancia en el cuarto 1 (cm)	0,807375		distancia en el cuarto 1 (cm)	0,67535	
9	distancia en el cuarto 2 (cm)	1,66865		distancia en el cuarto 2 (cm)	1,689475		distancia en el cuarto 2 (cm)	1,515375	
10	distancia en el cuarto 3 (cm)	1,980225		distancia en el cuarto 3 (cm)	2,247275		distancia en el cuarto 3 (cm)	2,3502	
11	distancia en el cuarto 4 (cm)	0,740025		distancia en el cuarto 4 (cm)	1,539175		distancia en el cuarto 4 (cm)	2,281375	
12	distancia total (cm)	5,554675		distancia total (cm)	6,2833		distancia total (cm)	6,8223	
13	ha ido más rápido	3		ha ido más rápido	3		ha ido más rápido	3	
14	ha ido más lento	4		ha ido más lento	1		ha ido más lento	1	
15	velocidad media	4,145279851		velocidad media	7,140113636		velocidad media	8,527875	
16	velocidad en el cuarto 1 (cm/s)	3,479925373		velocidad en el cuarto 1 (cm/s)	3,669886364		velocidad en el cuarto 1 (cm/s)	3,37675	
17	velocidad en el cuarto 2 (cm/s)	4,981044776		velocidad en el cuarto 2 (cm/s)	7,679431818		velocidad en el cuarto 2 (cm/s)	7,576875	
18	velocidad en el cuarto 3 (cm/s)	5,911119403		velocidad en el cuarto 3 (cm/s)	10,21488636		velocidad en el cuarto 3 (cm/s)	11,751	
19	velocidad en el cuarto 4 (cm/s)	2,209029851		velocidad en el cuarto 4 (cm/s)	6,99625		velocidad en el cuarto 4 (cm/s)	11,406875	
20	aceleracion media	1,648529739		aceleracion media	7,950284091		aceleracion media	14,25859375	
21	aceleracion en el cuarto 1 (cm/s)	10,38783693		aceleracion en el cuarto 1 (cm/s)	16,68130165		aceleracion en el cuarto 1 (cm/s)	16,88375	
22	aceleracion en el cuarto 2 (cm/s)	4,480953442		aceleracion en el cuarto 2 (cm/s)	18,22520661		aceleracion en el cuarto 2 (cm/s)	21,000625	
23	aceleracion en el cuarto 3 (cm/s)	2,77634217		aceleracion en el cuarto 3 (cm/s)	11,52479339		aceleracion en el cuarto 3 (cm/s)	20,870625	
24	aceleracion en el cuarto 4 (cm/s)	-11,05101359		aceleracion en el cuarto 4 (cm/s)	-14,63016529		aceleracion en el cuarto 4 (cm/s)	-1,720625	
25	punto inicio	derecha arriba		punto inicio	izquierda arriba		punto inicio	izquierda arriba	
26	Pendiente Recta	0,043276937		Pendiente Recta	-48,98467433		Pendiente Recta	-1,566382782	
27	fuerza media al inicio	1,618507463		fuerza media al inicio	1,585227273		fuerza media al inicio	1,447317073	
28	fuerza media en el segundo	1,759701493		fuerza media en el segundo	1,700227273		fuerza media en el segundo	1,693170732	
29	fuerza media en el tercero	1,849552239		fuerza media en el tercero	1,721136364		fuerza media en el tercero	1,744390244	
30	fuerza media al final	1,84880597		fuerza media al final	1,711818182		fuerza media al final	1,685121951	
31	media Fuerza	1,763		media Fuerza	1,672		media Fuerza	1,639	
32									
33									
34	ejercicio	1		ejercicio	5		ejercicio	6	
35	Angulo recta	44,99		punto inicio	Izquierda Arriba		punto inicio	Izquierda Arriba	

Sheet | child12-20.04.21-1 | child12-20.04.21-2 | child12-20.04.21-3 | child12-20.04.21-4 | child12-20.04.21-5 | child12-20.04.21-6 | child12-20.04.21- ...

Ilustración 17 Hoja principal con resúmenes

3. Arquitectura de la aplicación

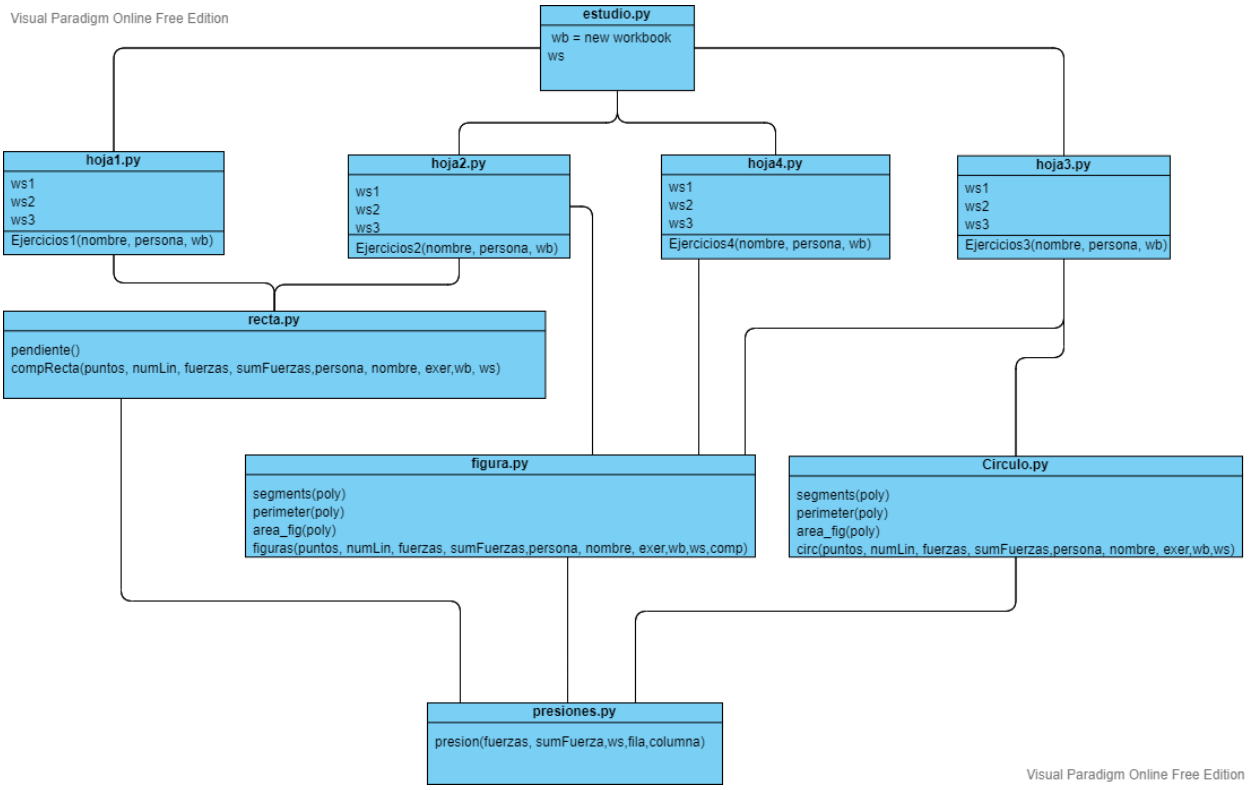


Ilustración 18 Diagrama de Clases

La aplicación está compuesta de nueve ficheros Python, *estudio.py* es el archivo principal, el cuál se encarga de llamar al resto de clases, crear el archivo Excel donde se almacenan todos los datos de la prueba realizada al alumno y posteriormente realizar el resumen de ellos, organizándolos y dándole el formato correspondiente.

Cada uno de los ficheros *hojax.py* está encargado de obtener los datos de cada una de las páginas dibujadas por los niños, es decir, de cada fichero *json*, realizando la lógica de obtención de dibujos en función de puntos *x* e *y*, asociándolos correctamente a cada tipo de dibujo que ha hecho. Cada uno de los ficheros está relacionado con una hoja, siendo hoja1.py la correspondiente con la hoja 1 del ejercicio realizado al niño. Es necesario que existan cuatro clases distintas para facilitar al usuario el uso de la aplicación, sin tener que modificar parámetros de estas como el nombre de cada fichero *json*, el Excel resultante, además de permitir dejar ejercicios sin rellenar en caso de que en esa prueba el alumno no haya completado todos los dibujos.

Como cada página se compone de dibujos de diferentes tipos (rectas, figuras geométricas planas, cruces y laberintos), cada fichero hoja llama a las funciones correspondientes para evaluar dichas pruebas. Además, incluye en el fichero Excel los datos de los puntos y las gráficas de velocidad,

aceleración y presión por cada trazo, junto con los datos evaluados para el ejercicio, por ejemplo, la hoja 1 llama únicamente a *recta.py*, mientras que la hoja 2 llama a *recta.py* y a *figura.py*.
Las clases de hoja reciben como parámetro el nombre del fichero (*childx-fecha.json*), el nombre del niño, y el fichero *wb* correspondiente con el Excel abierto por *estudio.py*. Además crea las variables *ws* correspondientes con cada una de las hojas que se escribirán en las clases a las que llama, aumentando la capacidad de reutilización de código.
El fichero *recta.py* es el encargado de evaluar las líneas, recibe como parámetro los datos correspondientes al alumno, los puntos y presiones, el número de trazos y los parámetros necesarios para completar el fichero Excel. Dentro de él se calcula la pendiente de la recta y la evaluación de los puntos recibidos según como se ha visto anteriormente en la explicación detallada del código. Además, escribe en la hoja Excel correspondiente con los datos evaluados que recibe como parámetro de la clase hoja, insertando valores analizados, los datos de los archivos y las gráficas.
Por otro lado, el fichero *figura.py* evalúa las cruces, figuras geométricas planas y los laberintos, recibiendo un comprobante (*booleano*) como parámetro para conocer si el dibujo que recibe corresponde con una figura que tenga superficie, en caso de que la tenga, se calculará el perímetro y el área, cosa que no tendría sentido evaluar para las cruces. Además, al igual que para recta.py, es necesario que reciba los datos del alumno, puntos y presiones, junto con los datos correspondientes al Excel para completar la hoja.
El siguiente fichero, *circulo.py*, se encarga de evaluar las circunferencias dibujadas por el alumno en la hoja número tres, tiene como añadido que calcula el índice de circularidad del dibujo, por ello no se podría reutilizar el fichero *figura.py*, además, las fórmulas empleadas para cálculo de áreas y perímetros son diferentes entre los tipos de dibujos. Como las anteriores funciones, recibe por parámetro los datos del alumno, los puntos y presiones y por último los datos necesarios para acceder y rellenar el Excel.
Por último, *presiones.py*, evalúa las presiones, viendo cuál es la media y en qué momentos del trazo ejerce más o menos presión. Como es una función empleada por todas las anteriores, sin importar el tipo de dibujo, permite la segmentación de la función del resto de ficheros, ahorrando tiempo de cómputo y mejorando los resultados obtenidos.
Es necesario enviar por parámetro las fuerzas ejercidas y la suma de ellas, para evitar volver a recorrerlas posteriormente, añadiendo innecesariamente más ciclos de ejecución. Además, igual que para recta, figura y círculo recibe *ws* y el nombre del fichero para escribir en el Excel una vez tenga evaluada la información captada por la tableta.

4. Presupuesto

Para hacer uso del Software correspondiente a la aplicación es necesario la adquisición de cierto material, a continuación, se enumera incluyendo un precio medio para la compra de todo lo necesario. Con los precios actualizados a junio de 2021:

1. Ordenador portátil …… 500€
2. Tableta gráfica Wacom bamboo Slate…….. 150€

El presupuesto medio es aproximadamente de 650€, en función del precio del portátil escogido, pudiendo reducir el coste final en caso de elegir un ordenador más básico, ya que no requiere un alto cómputo la ejecución de esta aplicación.

Respecto al tiempo que he empleado para realizar el trabajo se puede ver en el diagrama de Gantt inferior.

Actividad	Periodos							
	01-feb	08-feb	30-mar	15-abr	21-abr	23-may	09-jun	15-jun
Conocer ejercicios necesarios								
Análisis de diseño								
Programación								
Depuración de errores								
Analizar problemas con niños								
Adaptación de código								
Reunión para mostrar resultado								
Presentación del resultado final								

Ilustración 19 Diagrama de Gantt

5. Conclusiones

Este trabajo me ha servido para conocer una pequeña parte de la labor de los psicólogos, maestros de infantil y fisioterapeutas detectando problemas psicológicos y psicomotrices en niños de corta edad, llegando a entender lo importante que puede ser para el desarrollo de la persona la detección de uno de estos problemas en los primeros años de vida, pudiendo solucionarlo o buscando soluciones para que pueda vivir de la mejor forma posible con ese problema.

Esta aplicación que he creado facilitará el estudio de los ejercicios encargados de detectar ciertos problemas, ayudando a los profesionales a ver parámetros relevantes los cuales viendo simplemente el dibujo serían imposible de apreciar. Sirve para ayudar mejor al niño a superar los problemas relacionados con cada una de las características que aplique en su trazo.

Como trabajo futuro queda probar el software en colegios y experimentar todas las posibilidades de este en un entorno real, imposible de realizar actualmente debido a la condición social sanitaria que estamos viviendo hoy en día por causa del coronavirus.

Además, respecto a la hoja de los laberintos, el objetivo era conseguir que la aplicación reconociese de forma automática si el niño se salía o no del circuito marcado para dibujar. Debido a la limitación que supone el que la tableta gráfica no disponga de una pantalla, colocar un papel sobre ella con los laberintos dibujados y esperar que coincida exactamente con las esquinas para que lo evalúe correctamente resulta complicado, más teniendo en cuenta que la anchura de las calles del laberinto es del orden de 3 milímetros. Esto estaría resuelto fácilmente si se utiñizara una tableta gráfica con pantalla y se marcasen en ella las esquinas del papel, tratando de alinearlas para que coincida con los píxeles asignados mediante software al laberinto.

6. Bibliografía

1. ⊳ Test Beery- VMB -Evaluación De La Percepción Visual - Psicorevista. https://psicorevista.com/pruebas-psicometricas/test-beery-vmb-evaluacion-de-la-percepcion-visual/

2. Román RL, Resumen O. *Beery Buktenica Developmental Test of Visual Motor Integration (Beery VMI): Un Estudio de Comparación y Correlación*. Vol 1.; 2006. http://paideia.uprrp.edu

3. GitHub - gjiovi/pywillparser: Python 3 library to convert Wacom .will files in SVG or InkML. https://github.com/gjiovi/pywillparser

4. Regresión lineal - Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Regresión_lineal#El_modelo_de_regresión_lineal

5. Cómo Calcular Momentos de Hu en OpenCV - DataSmarts Español. https://datasmarts.net/es/como-calcular-momentos-de-hu-en-opencv/

6. Curso sobre como trabajar con hojas de cálculo (Excel, Calc) usando openpyxl en Python (VI) – Pybonacci. https://pybonacci.org/2021/04/07/curso-sobre-como-trabajar-con-hojas-de-calculo-excel-calc-usando-openpyxl-en-python-vi/

7. Charts — openpyxl 3.0.7 documentation.https://openpyxl.readthedocs.io/en/stable/charts/introduction.html

8. openpyxl - ajustar el tamaño del ancho de la columna. https://qastack.mx/programming/13197574/openpyxl-adjust-column-width-size

9. Listar directorio en Python. Listar ficheros de un directorio. https://j2logo.com/python/listar-directorio-en-python/

10. R. Peña. Resolución de problemas para ingenieros con Python estructurado. España: Ibergarceta Publicaciones S.L, 2016.

11. A. Montejo. Curso de Programación Python (MANUALES IMPRESCINDIBLES). España: Grupo Anaya Publicaciones Generales, 2019.

12. A. Sweigart. Automate The Boring Stuff With Python. No Starch Press,US, 2019.

13. A. M. Muñoz. Aprende Python en un fin de semana. España: Independently published, 2018

7. Anexos y/o Apéndices

Incluir gráficas dentro de un fichero Excel

Mediante *Openpyxl* es posible realizar la inserción de gráficas de distintos estilos (lineares, barras, etc) dentro de una hoja en un fichero Excel.
Es necesario aclarar que los objetos empleados para generar gráficas son únicos por gráfica, lo que significa que no se puede reutilizar la variable con el fin de añadir datos nuevos, eliminar algunos de ellos o modificar los valores.
Lo primero que hay que hacer es crear el objeto, a continuación se especifica cómo sería para una gráfica lineal, en caso de querer que sea un gráfico de barras habría que emplear el constructor *BarChart()*, igual que para el resto de estilos.

```
Grafico = LineChart()
```

Tras esto se debe elegir dónde se va a insertar, es decir, en qué hoja del fichero Excel. Para ello se asocia a una variable genérica *ws*:

```
ws = wb.create_sheet(nombre+"-1")
```

En caso de querer elegir una hoja ya existente en vez de crear una nueva se sustituye con el código siguiente:

```
ws = wb.get_sheet_by_name(nombre+"-1")
```

A continuación, hay que seleccionar los valores que se quieren insertar, para ello se emplea la función *Reference*:

```
values = Reference(ws,min_col = 4, max_col=4,min_row=filax1-numPunt, max_row=filax1)
```

Esto dice la hoja en la que se eligen los datos, la columna y la fila, por ejemplo si queremos que coja las columnas A y B de la hoja ws y las filas de la 1 a la 20, el código será:

```
values = Reference(ws,min_col = 1, max_col=2,min_row=1, max_row=20)
```

Con esta referencia se asocian los valores al gráfico, el código que lo consigue es:

```
grafico.add_data(values, titles_from_data=True)
```

Donde es posible decidir si se quiere que se muestre una leyenda con el nombre de los datos encontrados en las columnas de referencia.

Por último, se elige la celda en la que se desea insertar y se agrega el gráfico, siendo esta la correspondiente a la esquina superior izquierda:

celda = "M"+str(filax1-numPunt)
ws.add_chart(grafico,celda)

En el código de la aplicación de este trabajo se resta el número de puntos para que se escriba en la fila 2, es posible sustituir *filax1-numPunt* por el número de la fila donde se desea insertar.

Manual de usuario, instalación y mantenimiento

Lo primero que hace falta para utilizar el código del trabajo es tener instalado Python en su versión 3.0 o posterior, para ello se podrá descargar el ejecutable desde la página oficial de Python o desde la Windows Store, en función del sistema operativo que se corresponda con el equipo que va a ser utilizado para realizar el análisis de la prueba VMI.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		364158b3113cf8ac8db7868ce40ebc7b	25627989	SIG
XZ compressed source tarball	Source release		71f7ada6bec9cdbf4538adc326120cfd	19058600	SIG
macOS 64-bit Intel installer	Mac OS X	for macOS 10.9 and later	870e851eef2c6712239e0b97ea5bf407	29933848	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon	59aedbc04df8ee0547d3042270e9aa57	37732597	SIG
Windows embeddable package (32-bit)	Windows		cacf28418ae39704743fa790d404e6bb	7594314	SIG
Windows embeddable package (64-bit)	Windows		0b3a4a9ae9d319885eade3ac5aca7d17	8427568	SIG
Windows help file	Windows		b311674bd26a602011d8baea2381df9e	8867595	SIG
Windows installer (32-bit)	Windows		b29b19a94bbe498808e5e12c51625dd8	27281416	SIG
Windows installer (64-bit)	Windows	Recommended	53a354a15baed952ea9519a7f4d87c3f	28377264	SIG

Ilustración 20 Descarga de Python

Una vez descargado el fichero correspondiente, se deberá proceder a la instalación, al acabar, se ejecuta el programa comprobando que funciona correctamente y obteniendo la pantalla de comandos de Python:

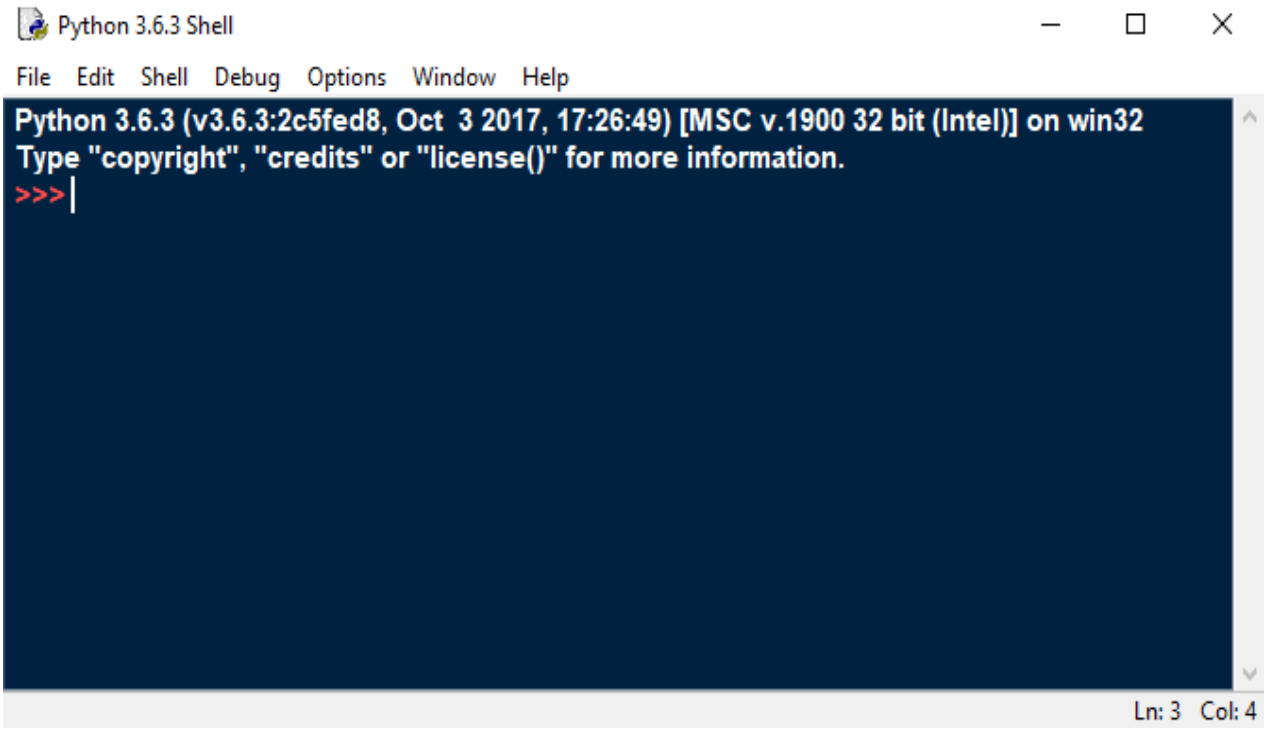


Ilustración 21 Python iniciado

Tras realizar la instalación, se deberán agregar los paquetes y librerías necesarias para la correcta ejecución del programa, lo primero será instalar el módulo *pip*:

1. Descarga del *script* del instalador *get-pip.py* correspondiente a la versión Python descargada anteriormente.
2. A continuación, se abrirá el terminal de Windows (tecla Windows + r, escribir "*cmd*" y apretar a ejecutar), allí se navegará hasta el archivo get-pip.py previamente descargado (mediante el comando cd):

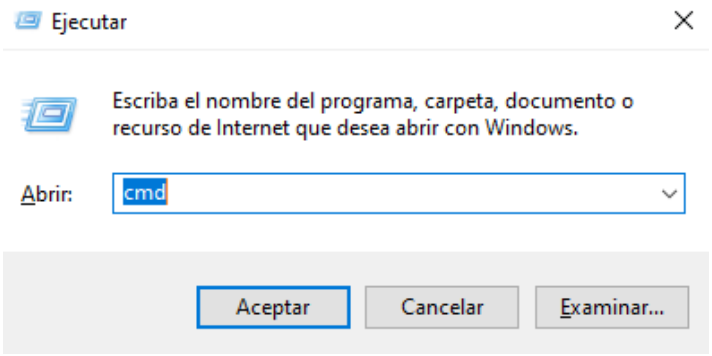


Ilustración 22 Acceso al terminal

3. Se deberá ejecutar el comando *python get-pip.py*.

Este comando descarga el paquete *pip*, el cual permite descargar complementos extra para Python, como los paquetes que es necesario importar.
Es necesario ejecutar los siguientes comandos dentro del terminal:

```
>>py -m pip install openpyxl
>>py -m pip install json
>>py -m pip install matplotlib
>>py -m pip install sklearn
```

Los cuales añaden *Openpyxl*, *json*, *matplotlib* y *sklearn* al directorio de Python, permitiendo su uso en las diferentes clases que componen esta aplicación.
Una vez configurado Python es posible comenzar a realizar la prueba VMI a los niños, en caso de que la tableta gráfica no esté vinculada con el portátil, se realizará la conexión como indica en la aplicación *Inkspace* descargada desde la página oficial de Wacom.
Con la tableta ya vinculada, colocará de forma horizontal con el botón lo más cerca posible del niño, se colocará sobre ella el cuadernillo de ejercicios dejando la página que se va a rellenar en contacto directo con la tableta, como se muestra a continuación:

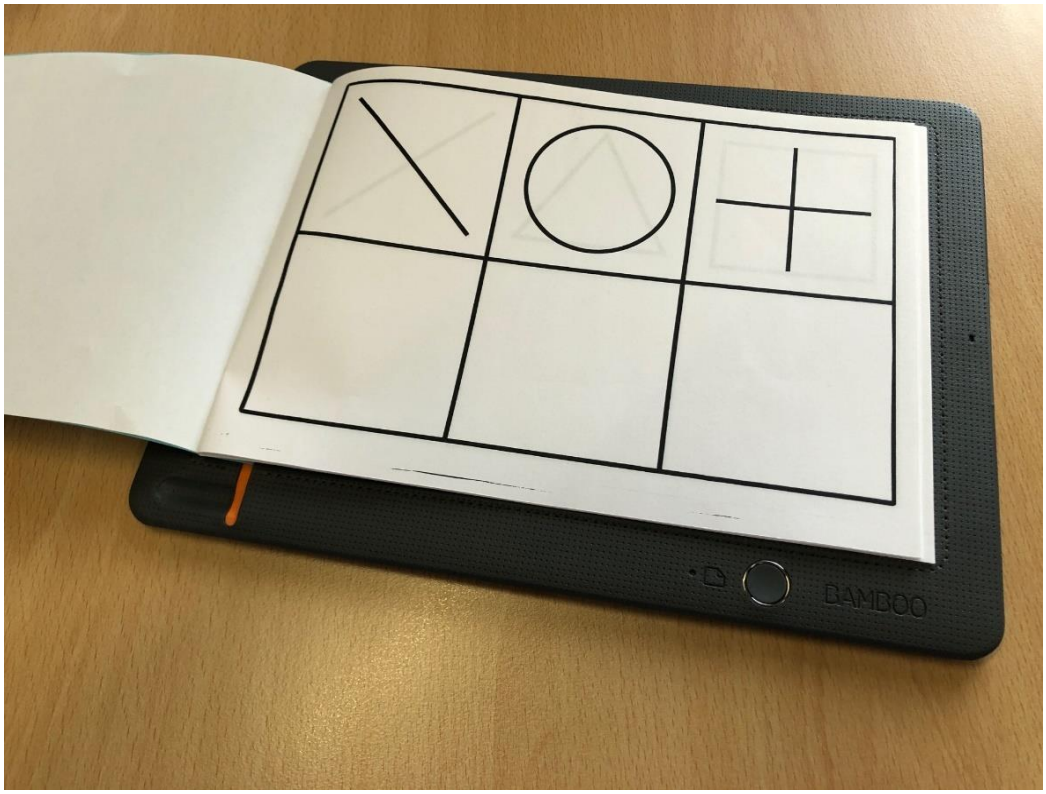


Ilustración 23 Forma de colocar el cuadernillo

Se le dará al niño el bolígrafo que viene con la tableta gráfica, ya que es el único elemento de entrada de texto que detecta (con un bolígrafo normal o un lápiz no funciona), cuando el alumno termine de dibujar los ejercicios que tiene frente a él se pulsará en el botón del dispositivo, viendo como el led pasa de color azul a verde, significando que ya ha guardado el dibujo en la memoria interna y está preparado para sincronizarse con el ordenador. Esto deberá hacerse para cada una de las hojas que se quieran evaluar posteriormente.
Se accederá a la aplicación *Inkspace* y se esperará hasta que aparezcan los dibujos realizados en la pantalla de inicio, se accederá a cada una de estas páginas y se pinchará en la opción de compartir el fichero, seleccionando el formato Will. A la hora del guardado se seguirá el siguiente formato:
"child" + número niño + "-" + fecha del día en el que se realiza + ".-" + ejercicio ".json"
Siendo ejercicio el número de la hoja que está rellenando, al haber cuatro hojas, para la primera hoja del niño 34, en el día dieciocho de mayo de 2021, será el resultado del archivo a exportar:

- *child34-18.05.21.-1.will*

Tras esto, se accede al programa *Pywillparser* descargado previamente, pegando los archivos obtenidos anteriormente, a la altura de "test.py", se accederá al fichero Python y se escribirá lo siguiente para cada archivo Will generado:

wp.open("child34-18.05.21.-1.will')

wp.save_as_json('output/child34-18.05.21.-1.json')

Esto abre el archivo que acaba de ser generado y se guarda como *json*.
Una vez se tengan todos los archivos convertidos, se copiarán en la carpeta "*data*" del programa del trabajo:

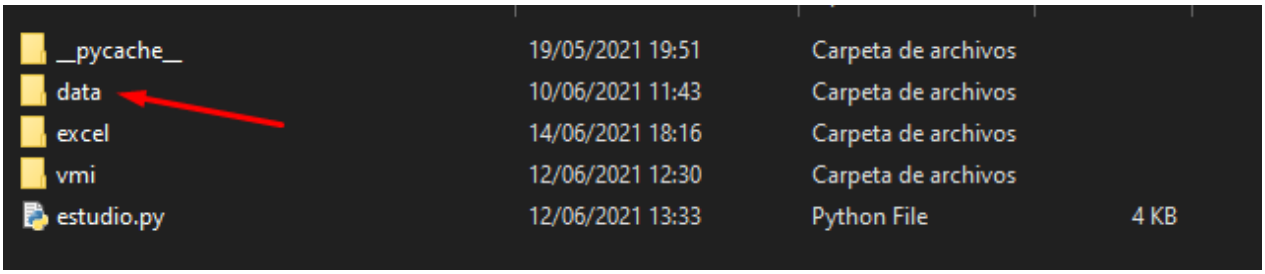


Ilustración 24 Directorio de la aplicación

A continuación, se accede al fichero "*estudio.py*" (click derecho y *Edit with Idle*), el cuál ejecuta la aplicación encargada de evaluar los dibujos y exportar los resultados a un Excel.

Con esto ya es posible ejecutar la aplicación, para ello se pulsará "*F5*" dentro de "*estudio.py*", o en la barra superior, pulsar Run y posteriormente Run module.

Tras unos segundos aparecerá una nueva ventana de Python donde se muestra la ejecución del programa:

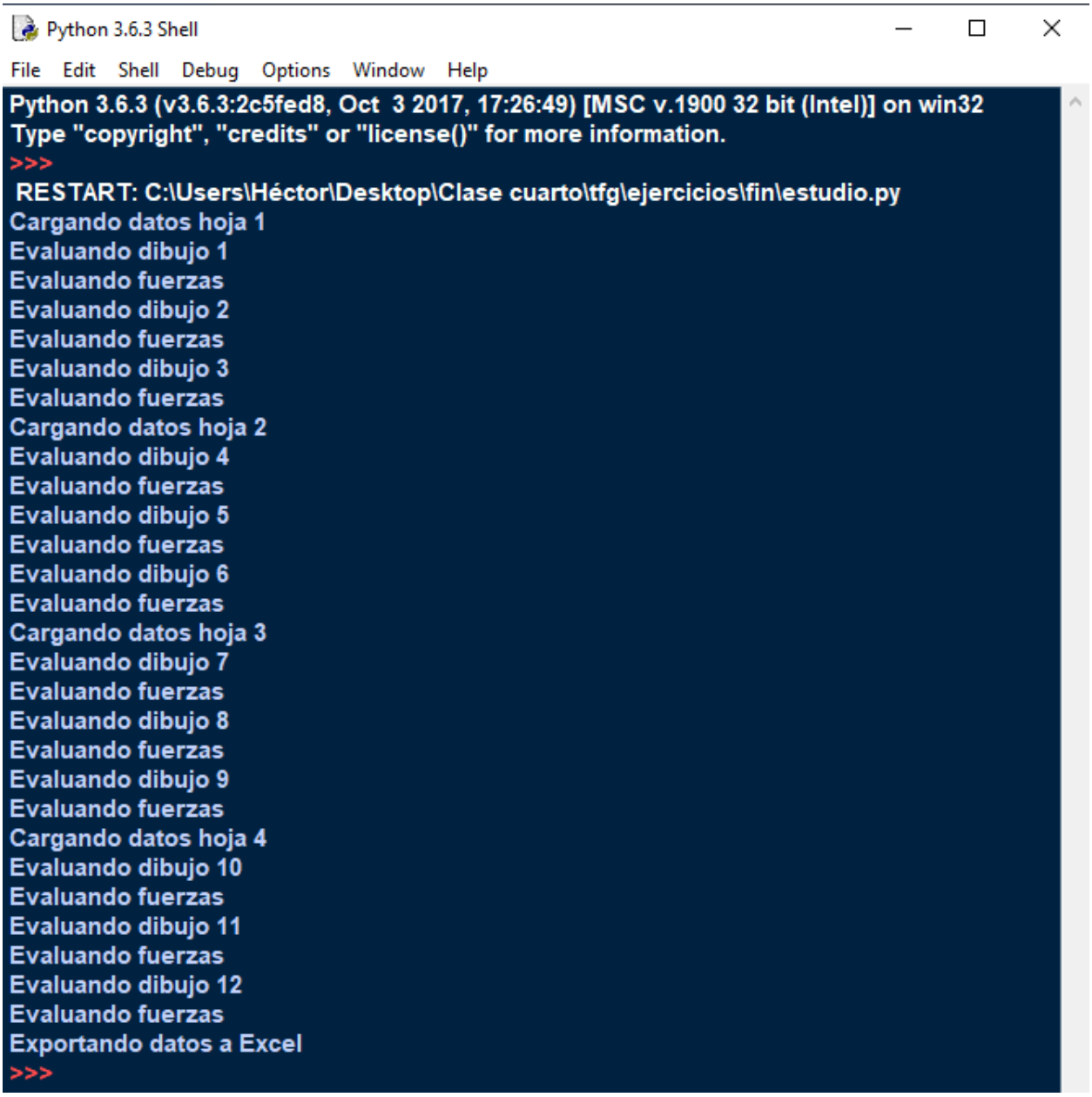


Ilustración 25 Ejecución de la aplicación

Al acabar la ejecución del programa, dentro del directorio de la aplicación, en la carpeta Excel se encuentra el archivo xlsx correspondiente a los datos evaluados anteriormente, manteniendo el nombre puesto en fichero *json*:

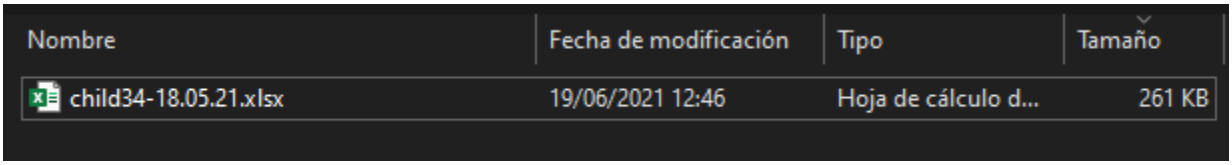


Ilustración 26 Excel dentro de carpeta

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR